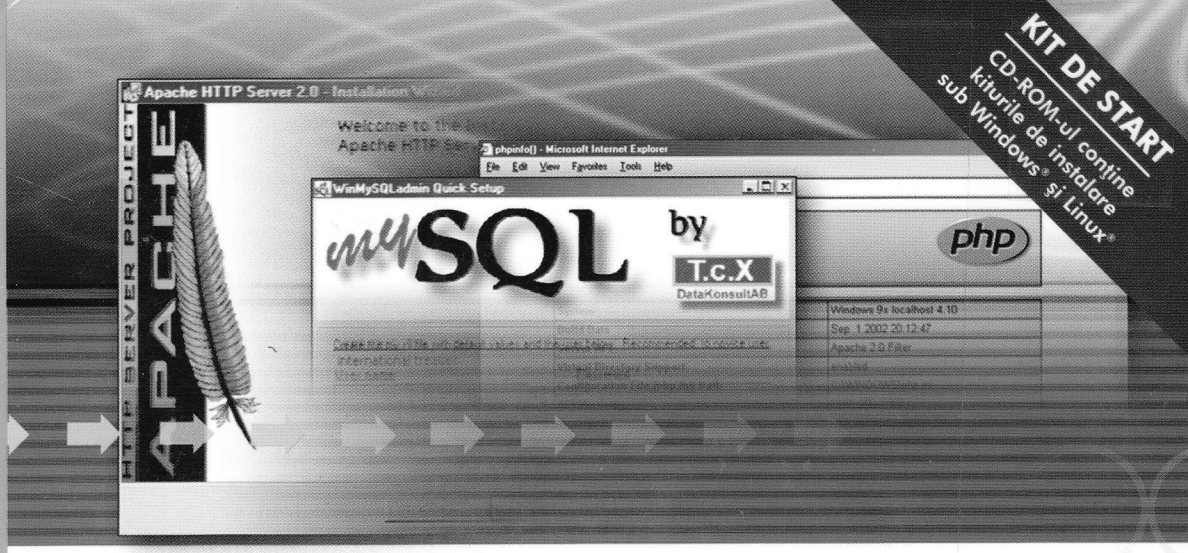


Julie C. Meloni

Învață singur

PHP, MySQL® și Apache

Toate într-o singură carte

CORINT

CD-ROM INCLUS

Despre autori

Autorul principal

Julie C. Meloni este directorul tehnic al i2i Interactive (`http://www.i2ii.com`), o companie multimedia din Los Altos, California. Ea a dezvoltat aplicaţii Web încă de la începuturile epocii Web. Îşi aminteşte, legat de aceasta, starea de efervescenţă creată în momentul apariţiei primului browser Web cu interfaţă grafică. Julie Meloni este autoarea mai multor cărţi şi articole despre limbaje de programare pentru Web şi despre baze de date, unele traduse în mai multe limbi, inclusiv în chineză, italiană, portugheză, poloneză şi chiar în sârbă.

Autori care au colaborat la carte

Matt Zandstra este scriitor tehnic şi consultant specializat în programarea la nivel de server. Împreună cu partenerul său de afaceri, Max Guglielmino, conduce firma Corrosive (`http://www.corrosive.co.uk`), o agenţie tehnică al cărei obiect de activitate este planificarea, proiectarea şi construirea de aplicaţii Internet. Interesat de toate aspectele programării orientate pe obiecte, Matt explorează, în momentul de faţă, modele de proiectare de aplicaţii în PHP 5. Când nu citeşte, nu scrie şi nu se gândeşte la scrierea de cod în PHP şi Java, el împuşcă invadatori extratereştri în parc cu fiica sa de patru ani, Holly. Împreună cu partenera sa, Louise McDougall, şi cu copiii lor, Holly şi Jake, Matt locuieşte pe malul mării.

Daniel López Ridruejo este fondatorul companiei BitRock, companie din domeniul tehnologiei informaţiei care oferă instrumente de instalare multiplatformă şi de administrare pentru o gamă largă de produse, atât comerciale, cât şi de tip open source. Înainte de aceasta, el a făcut parte din echipa iniţială de ingineri de la Covalent Technologies, Inc., echipă care punea la dispoziţia companiilor produsul software Apache, asistenţă tehnică şi service la nivel de corporaţie. Este autorul câtorva ghiduri foarte populare de Apache şi Linux, autorul modulului `mod_mono`, care integrează serverul Apache cu complexul .NET, şi al lui Comanche, un instrument vizual de configurare pentru Apache. Daniel este un orator invitat permanent la conferinţe open source, cum ar fi Linux World, ApacheCon, precum şi O'Reilly Open Source Convention. A obţinut diplomă de Master în telecomunicaţii în cadrul Escuela Superior de Ingenieros de Sevilla (Şcoala Superioară de Inginerie din Sevilla) şi a Danmarks Tekniske Universitet (Universitatea Tehnică din Danemarca). Daniel este membru al Apache Software Foundation.

Cuprins pe scurt

Introducere 1

PARTEA I — Elemente fundamentale

Capitolul 1 Instalarea şi configurarea serverului MySQL 7

2 Instalarea şi configurarea serverului Apache 27

3 Instalarea şi configurarea pachetului PHP 51

PARTEA a II-a — Structura limbajului PHP

Capitolul 4 Elementele constitutive ale limbajului PHP 71

5 Funcţii de controlare a execuţiei codului în PHP 93

6 Folosirea funcţiilor 115

7 Lucrul cu vectorii şi obiectele 135

PARTEA a III-a — La lucru cu codul

Capitolul 8 Lucrul cu tipurile şir de caractere, dată şi oră 153

9 Folosirea formularelor 181

10 Lucrul cu sesiunile de utilizator 203

11 Manipularea fişierelor şi directoarelor 213

12 Lucrul cu mediul de operare al serverului 237

13 Manipularea imaginilor 245

PARTEA a IV-a — Integrarea PHP cu MySQL

Capitolul 14 Procesul de proiectare a bazelor de date 261

15 Comenzi SQL de bază 273

16 Interacţiunea între scripturile PHP şi MySQL 321

PARTEA a V-a — Proiecte fundamentale

Capitolul 17 Gestionarea unei liste de e-mail simple 335

18 Crearea unei agende de adrese online 347

19 Crearea unui forum de discuţii simplu 371

20 Crearea interfeţei pentru un magazin virtual 387

21 Crearea unui mecanism pentru un coş de cumpărături virtual 399

22 Crearea unui calendar simplu 413

23 Restricţionarea accesului la aplicaţiile dumneavoastră 427

24 Înregistrarea şi monitorizarea activităţii serverului Web 449

25 Localizarea aplicaţiilor 467

PARTEA a VI-a — Administrare şi optimizare

Capitolul 26 Reglarea performanţei serverului Apache şi folosirea host-urilor virtuale 481

27 Configurarea unui server Web securizat 499

28 Optimizarea şi reglarea serverului MySQL 515

29 Actualizări de software 533

PARTEA a VII-a — Gânduri spre viitor

Capitolul 30 Facilităţile noi şi compatibilitatea cu versiunile anterioare ale PHP 5.0 541

31 Facilităţile noi şi compatibilitatea cu versiunile anterioare ale MySQL 4.1 547

PARTEA a VIII-a — Anexe

Anexa A Instalarea aplicaţiilor MySQL, Apache şi PHP aflate pe CD-ROM-ul ataşat cărţii 557

Index 569

Cuprins

Introducere 1

Cine ar trebui să citească această carte 1

Cum este organizată această carte 2

Convenţii utilizate în carte 3

PARTEA I — Elemente fundamentale

Capitolul 1. Instalarea şi configurarea serverului MySQL 7

Versiuni curente şi viitoare de MySQL 7

Cum găsiţi kitul de instalare MySQL 8

Instalarea MySQL în cadrul sistemului de operare Linux/Unix 8

Instalarea MySQL în cadrul sistemului de operare Windows 9

Depanarea procesului de instalare 16

Considerente fundamentale de securitate 16

Pornirea serverului MySQL 17

Securizarea conexiunii la MySQL 18

Introducere în sistemul de privilegii al MySQL 18

Procesul de autentificare în doi paşi 19

Lucrul cu privilegiile acordate utilizatorilor 20

Adăugarea utilizatorilor 21

Revocarea privilegiilor 23

Rezumat 23

Întrebări şi răspunsuri 25

Atelier 25

Chestionar 25

Răspunsuri 25

Activităţi 26

Capitolul 2. Instalarea şi configurarea serverului Apache 27

Versiuni curente şi viitoare ale serverului Apache 27

Alegerea unei metode de instalare 28

Compilarea surselor 28

Instalarea folosind fişiere binare 28

Instalarea serverului Web Apache pe platforma Linux/Unix 29

Descărcarea codurilor sursă a serverului Apache ... 29
Decomprimarea codurilor sursă a serverului Web Apache ... 29
Pregătiri pentru instalare ... 30
Compilarea şi instalarea serverului Apache ... 31
Instalarea serverului Web Apache sub sistemul de operare Windows ... 32
Structura fişierului de configurare a serverului Apache ... 35
Directivele ... 35
Containerele ... 37
Evaluarea condiţiilor ... 38
Directiva `ServerRoot` ... 39
Fişierele de configurare per director ... 40
Fişierele jurnal ale serverului Apache ... 41
Fişierul `access_log` ... 41
Fişierul `error_log` ... 41
Fişiere suplimentare ... 42
Comenzi ce pot fi trimise serverului Apache ... 42
Fişierul executabil al serverului Apache ... 42
Scriptul de control al serverului Apache ... 43
Prima pornire a serverului Apache ... 44
Verificaţi-vă fişierul de configurare ... 44
Iniţializarea serverului Apache ... 45
Depanare ... 46
Un server preexistent ... 46
Nu aveţi permisiunea să alocaţi portul ... 46
Acces nepermis ... 46
Configurări de grup greşite ... 47
Rezumat ... 47
Întrebări şi răspunsuri ... 48
Atelier ... 48
Test ... 48
Soluţii ... 49
Activităţi ... 49

Capitolul 3. Instalarea şi configurarea pachetului PHP ... 51

Versiunea curentă şi versiunile viitoare de PHP ... 51
Compilarea pachetului PHP pentru Apache sub Linux/Unix ... 52

Opţiuni suplimentare de configurare 53
Integrarea dintre PHP şi Apache sub Linux/Unix 54
Instalarea fişierelor PHP sub Windows 55
Integrarea dintre PHP şi Apache sub Windows 55
Elementele de bază ale fişierului `php.ini` 56
Verificarea instalării 57
Obţinerea de asistenţă pentru instalare 58
Elementele de bază ale scripturilor PHP 59
Deschiderea şi închiderea blocurilor cu instrucţiuni PHP 60
Instrucţiunea `echo` şi funcţia `print()` 62
Combinarea codului HTML cu cod PHP 63
Adăugarea de comentarii în codul dumneavoastră PHP 64
Rezumat 65
Întrebări şi răspunsuri 66
Atelier 66
Test 66
Soluţii 67
Activităţi 67

PARTEA a II-a — Structura limbajului PHP

Capitolul 4. Elementele constitutive ale limbajului PHP 71

Variabilele 71
Variabile globale şi superglobale 73
Tipuri de date 74
Schimbarea tipului prin folosirea funcţiei `settype()` 76
Schimbarea tipului prin casting (conversia valorii variabilei) 77
De ce să verificăm tipul de date? 79
Operatori şi expresii 79
Operatorul de atribuire 80
Operatori aritmetici 81
Operatorul de concatenare 81
Operatorii combinaţi de atribuire 82
Incrementarea şi decrementarea automată a unei variabile de tip `integer` 83
Operatori de comparare 84
Crearea unor expresii condiţionale mai complexe cu ajutorul operatorilor logici 85

Ordinea operaţiilor ... 86
Constante ... 87
Constante predefinite ... 89
Rezumat ... 89
Întrebări şi răspunsuri ... 90
Atelier ... 90
Test ... 90
Soluţii ... 91
Activităţi ... 92
Capitolul 5. Funcţii de controlare a execuţiei codului în PHP ... 93
Alternarea codului de executat ... 93
Instrucţiunea `if` ... 94
Folosirea clauzei `else` împreună cu instrucţiunea `if` ... 95
Folosirea clauzei `else if` împreună cu instrucţiunea `if` ... 96
Instrucţiunea `switch` ... 97
Folosirea operatorului `?` ... 99
Structuri repetitive ... 100
Instrucţiunea `while` ... 100
Instrucţiunea `do...while` ... 101
Instrucţiunea `for` ... 102
Părăsirea buclelor cu ajutorul instrucţiunii `break` ... 104
Oprirea unei singure iteraţii cu ajutorul instrucţiunii `continue` ... 106
Imbricarea structurilor repetitive ... 107
Blocurile de cod şi datele afişate în browser ... 109
Rezumat ... 110
Întrebări şi răspunsuri ... 112
Atelier ... 112
Test ... 112
Soluţii ... 113
Activitate ... 113
Capitolul 6. Folosirea funcţiilor ... 115
Ce este o funcţie? ... 115
Apelarea funcţiilor ... 116
Definirea unei funcţii ... 117
Valorile returnate de către funcţiile definite de utilizator ... 120
Domeniul de definire al variabilelor ... 121

Accesarea variabilelor cu instrucţiunea `global` 122
Păstrarea stării între apelările unei funcţii prin folosirea instrucţiunii `static` 124
Mai multe despre argumente 126
Stabilirea valorilor implicite ale argumentelor 127
Pasarea de referinţe către funcţii 128
Verificarea existenţei unei funcţii 130
Rezumat 132
Întrebări şi răspunsuri 133
Atelier 133
Test 133
Soluţii 134
Activitate 134

Capitolul 7. Lucrul cu vectorii şi obiectele 135
Ce este un vector? 135
Crearea vectorilor 135
Crearea vectorilor asociativi 137
Crearea vectorilor multidimensionali 137
Câteva funcţii pentru lucrul cu vectori 139
Crearea unui obiect 140
Proprietăţile obiectelor 142
Metodele obiectelor 143
Constructori 145
Moştenirea obiectelor 145
Rezumat 147
Întrebări şi răspunsuri 148
Atelier 148
Test 148
Soluţii 148
Activităţi 149

PARTEA a III-a — La lucru cu codul

Capitolul 8. Lucrul cu tipurile şir de caractere, dată şi oră 153
Formatarea şirurilor de caractere în PHP 154
Folosirea funcţiei `printf()` 154

Funcţia `printf()` şi specificatorii de tip 155
Completarea rezultatului cu caractere „de umplutură" prin folosirea specificatorilor de completare 156
Specificarea lăţimii unui câmp 157
Specificarea preciziei 158
Specificaţiile de conversie: recapitulare 159
Schimbarea poziţiei argumentelor 161
Stocarea unui şir de caractere formatat 162
Analizarea şirurilor în PHP 162
O notă despre indexarea şirurilor 162
Aflarea lungimii unui şir cu ajutorul funcţiei `strlen()` 163
Găsirea unui subşir în interiorul unui şir cu ajutorul funcţiei `strstr()` 163
Aflarea poziţiei unui subşir cu ajutorul funcţiei `strpos()` 163
Extragerea unei părţi a unui şir cu ajutorul funcţiei `substr()` 164
Împărţirea unui şir în fragmente cu ajutorul funcţiei `strtok()` 165
Manipularea şirurilor în PHP 166
Curăţarea unui şir cu ajutorul funcţiilor `trim()` şi `ltrim()` şi `strip_tags()` 166
Înlocuirea unei porţiuni a unui şir cu ajutorul funcţiei `substr_replace()` 168
Înlocuirea subşirurilor cu ajutorul funcţiei `str_replace()` 168
Schimbarea capitalizării literelor 169
Ajustarea lungimii rândurilor cu ajutorul funcţiilor `wordwrap()` şi `nl2br()` 170
Împărţirea şirurilor în vectori cu ajutorul funcţiei `explode()` 172
Folosirea funcţiilor PHP dedicate datelor calendaristice şi orelor 172
Obţinerea datei curente cu ajutorul funcţiei `time()` 172
Convertirea unui *timestamp* cu ajutorul funcţiei `getdate()` 173
Convertirea unui *timestamp* folosind funcţia `date()` 174
Crearea de *timestamp*-uri cu ajutorul funcţiei `mktime()` 176
Testarea unei date cu ajutorul funcţiei `checkdate()` 178
Rezumat 178
Atelier 179
Întrebări şi răspunsuri 179
Test 179
Soluţii 180
Activităţi 180

Capitolul 9. Folosirea formularelor 181

Crearea unui formular simplu de introducere a datelor 181

Accesarea datelor introduse în formular folosind vectori definiţi de utilizator 183

Combinarea codului HTML şi PHP pe o singură pagină 186

Folosirea câmpurilor ascunse pentru păstrarea stării 189

Redirecţionarea utilizatorului 190

Trimiterea unui mesaj e-mail la primirea formularului 191

Configurarea sistemului pentru funcţia `mail()` 192

Crearea formularului 193

Crearea scriptului care să trimită mesajul e-mail 194

Formatarea mesajului e-mail cu HTML 196

Încărcarea fişierelor pe server 197

Crearea formularului de încărcare a fişierelor 198

Crearea scriptului de încărcare a fişierelor 199

Rezumat 201

Atelier 202

Test 202

Soluţii 202

Activităţi 202

Capitolul 10. Lucrul cu sesiunile de utilizator 203

Privire de ansamblu asupra funcţiilor dedicate sesiunilor 203

Iniţierea unei sesiuni 204

Folosirea variabilelor de sesiune 205

Pasarea ID-urilor de sesiune prin intermediul şirurilor de interogare 209

Distrugerea sesiunilor şi dealocarea variabilelor 210

Rezumat 211

Întrebări şi răspunsuri 212

Atelier 212

Test 212

Soluţii 212

Activitate 212

Capitolul 11. Manipularea fişierelor şi directoarelor 213

Includerea fişierelor cu ajutorul instrucţiunii `include()` 213

Returnarea unei valori dintr-un document inclus 215

Folosirea instrucţiunii include() în cadrul structurilor de control 216
Instrucţiunea include_once() 217
Directiva include_path 217
Verificarea fişierelor 218
Verificarea existenţei unui fişier cu ajutorul funcţiei file_exists() 218
Este fişier sau director? 218
Verificarea stării unui fişier 219
Determinarea dimensiunii unui fişier cu ajutorul funcţiei filesize() 219
Obţinerea informaţiilor despre data fişierului 220
Crearea unei funcţii care efectuează mai multe verificări asupra fişierelor 220
Crearea şi ştergerea fişierelor 222
Deschiderea unui fişier pentru scriere, citire sau adăugare 222
Citirea din fişiere 223
Citirea liniilor dintr-un fişier cu fgets() şi feof() 224
Citirea unui volum arbitrar de date dintr-un fişier cu ajutorul funcţiei fread() 225
Citirea de caractere dintr-un fişier cu ajutorul funcţiei fgetc() 227
Scrierea sau adăugarea într-un fişier 228
Scrierea într-un fişier cu ajutorul funcţiei fwrite() sau fputs() 229
Blocarea fişierelor cu ajutorul funcţiei flock() 230
Manipularea directoarelor 231
Crearea directoarelor cu funcţia mkdir() 231
Ştergerea unui director cu ajutorul funcţiei rmdir() 231
Deschiderea unui director pentru citire cu opendir() 232
Citirea conţinutului unui director cu ajutorul funcţiei readdir() 232
Rezumat 234
Întrebări şi răspunsuri 235
Atelier 235
Test 235
Soluţii 236
Activităţi 236

Capitolul 12. Lucrul cu mediul de operare al serverului 237

Deschiderea de canale de comunicare (pipes) către şi dinspre procese folosind funcţia popen() 237

Executarea de comenzi cu ajutorul funcţiei `exec()` 239
Executarea comenzilor folosind `system()` sau `passthru()` 241
Rezumat 242
Întrebări şi răspunsuri 243
Atelier 243
Test 243
Soluţii 243

Capitolul 13. Manipularea imaginilor 245

Să înţelegem procesul de creare a unei imagini 245
Câteva cuvinte despre culoare 245
Modificările ce trebuie făcute instalării de PHP 246
Obţinerea bibliotecilor suplimentare 247
Desenarea unei imagini noi 247
Desenarea formelor şi a liniilor 248
Umplerea suprafeţelor cu culoare 250
Înfrumuseţarea paginilor cu ajutorul diagramelor circulare 251
Modificarea imaginilor deja existente 254
Rezumat 257
Întrebări şi răspunsuri 258
Atelier 258
Test 258
Soluţii 258
Activitate 259

PARTEA a IV-a — Integrarea PHP cu MySQL

Capitolul 14. Procesul de proiectare a bazelor de date 261

Importanţa proiectării corecte a bazelor de date 261
Tipuri de relaţii între tabele 262
Relaţii unul la unul 263
Relaţii de tip unul la mai multe 264
Relaţii de tipul mai multe la mai multe 265
Să înţelegem normalizarea 266
Problemele tabelei bidimensionale 267
Prima formă normală 268
Cea de-a doua formă normală 268

Cea de-a treia formă normală ... 269
Implementarea procesului de proiectare ... 270
Rezumat ... 271
Întrebări şi răspunsuri ... 272
Atelier ... 272
Test ... 272
Soluţii ... 272
Activitate ... 272
Capitolul 15. Comenzi SQL de bază ... 273
Tipurile de date în MySQL ... 273
Tipuri de date numerice ... 274
Tipuri calendaristice de date ... 275
Tipuri de şiruri de caractere ... 276
Sintaxa de creare a unei tabele ... 277
Folosirea comenzii INSERT ... 278
Să privim mai îndeaproape comanda INSERT ... 278
Folosirea comenzii SELECT ... 280
Ordonarea rezultatelor comenzii SELECT ... 281
Limitarea rezultatelor ... 282
Folosirea clauzei WHERE în interogările dumneavoastră ... 283
Folosirea operatorilor în clauzele WHERE ... 283
Comparaţiile între şiruri de caractere cu ajutorul operatorului LIKE ... 284
Selectarea înregistrărilor din mai multe tabele ... 285
Folosirea cuvântului cheie JOIN ... 287
Folosirea comenzii UPDATE pentru modificarea înregistrărilor ... 289
Instrucţiuni UPDATE condiţionale ... 291
Folosirea valorilor existente în coloane în cadrul instrucţiunilor UPDATE ... 292
Folosirea comenzii REPLACE ... 292
Folosirea comenzii DELETE ... 293
Ştergerea condiţională ... 294
Funcţii de manipulare a şirurilor de caractere folosite în mod uzual în MySQL ... 296
Funcţii referitoare la lungime şi funcţii de concatenare ... 296
Funcţii de ajustare şi completare ... 298

Funcţii de găsire şi poziţionare 300
Funcţii care operează cu subşiruri 300
Funcţii de modificare a şirurilor de caractere 301
Folosirea funcţiilor MySQL dedicate datelor calendaristice şi orelor 303
Operaţii cu zile 303
Operaţii cu luni şi ani 306
Operaţii cu săptămâni 307
Operaţii cu ore, minute şi secunde 308
Formatarea datelor calendaristice şi a orelor în MySQL 309
Calculele aritmetice cu date calendaristice în MySQL 311
Funcţii speciale şi operaţiuni de conversie 313
Rezumat 315
Întrebări şi răspunsuri 317
Atelier 318
Test 318
Soluţii 318
Activitate 319

Capitolul 16. Interacţiunea între scripturile PHP şi MySQL 321

Conectarea la MySQL folosind PHP 321
Folosirea funcţiei `mysql_connect()` 322
Executarea interogărilor 323
Preluarea mesajelor de eroare 324
Lucrul cu datele MySQL 325
Inserarea datelor prin intermediul PHP 325
Extragerea datelor în PHP 329
Alte funcţii PHP de lucru cu MySQL 331
Rezumat 331
Atelier 332
Test 332
Soluţii 332

PARTEA a V-a — Proiecte fundamentale

Capitolul 17. Gestionarea unei liste de e-mail simple 335

Dezvoltarea mecanismului de abonare 335
Crearea tabelei `subscribers` 336
Crearea formularului de abonare 336

Introducere

Bine ați venit! Lucrarea *Învață singur PHP, MySQL și Apache — Toate într-o singură carte* combină lecțiile pe care le puteți găsi în cărțile *Sams Teach Yourself Apache 2 in 24 Hours* (*Învață singur Apache 2 în 24 de lecții*), *Sams Teach Yourself PHP in 24 Hours* (*Învață singur PHP în 24 de lecții*) și *Sams Teach Yourself MySQL in 24 Hours* (*Învață singur MySQL în 24 de lecții*), toate trei apărute anterior sub egida aceleiași prestigioase edituri **SAMS.** Alături de ele, veți găsi câteva capitole adiționale, care vă oferă o introducere temeinică și lipsită de efort în lumea dezvoltării de aplicații Web, cu ajutorul acestor trei tehnologii.

În cadrul acestei cărți veți învăța conceptele necesare pentru a configura și administra serverul Web Apache, conceptele fundamentale ale programării PHP, precum și metode de utilizare și administrare a bazelor de date relaționale MySQL. Scopul principal al cărții este să vă ofere posibilitatea de a înțelege cât de bine și cum se integrează împreună aceste tehnologii, precum și de a vă prezenta o serie de cunoștințe practice despre utilizarea lor combinată în cadrul unui mediu de dezvoltare.

Cine ar trebui să citească această carte

Această carte este destinată tuturor celor care înțeleg la modul general conceptele utilizării unui mediu de dezvoltare a aplicațiilor bazate pe tehnologii Web, fie ele în contextul Linux/Unix sau în cadrul Windows. Lecțiile de instalare și configurare pleacă de la premisa unei familiarizări anterioare cu sistemul de operare folosit și cu metodele de bază de compilare (pe sistemele Linux/Unix) sau de instalare (pe sistemul de operare Windows) a diferitelor aplicații.

Lecțiile care tratează partea de programare folosind PHP nu presupun cunoștințe anterioare despre acest limbaj, dar experiența în folosirea altor limbaje de programare precum C sau Perl vă poate fi foarte folositoare pentru o mai ușoară înțelegere. De asemenea, experiența anterioară în utilizarea altor tipuri de baze de date precum Oracle sau Microsoft SQL Server nu este necesară, dar vă poate oferi un punct de plecare pentru a parcurge mai ușor lecțiile legate de MySQL.

Singura cerință, în adevăratul sens al cuvântului, este înțelegerea creării de pagini cu conținut static utilizând codul HTML. Dacă această carte reprezintă primul pas în lumea dezvoltării Web, o veți putea totuși folosi, dar ar fi util să parcurgeți mai întâi un material despre bazele limbajului HTML. Dacă nu aveți probleme în lucrul cu documentele și sunteți capabil să creați un tabel simplu în HTML, atunci vă veți descurca de minune.

Cum este organizată această carte

Cartea este divizată în şapte părţi, fiecare fiind asociată unui anumit subiect. Lecţiile din fiecare parte trebuie citite una după cealaltă, fiecare dintre ele construindu-se pe informaţiile discutate în lecţia anterioară:

- Partea I, *Elemente fundamentale*, vă prezintă pas cu pas procesele de instalare şi configurare ale MySQL, Apache şi PHP. Va trebui să parcurgeţi lecţiile din această primă parte înainte de a trece la programarea efectivă, cu excepţia cazului în care aveţi deja acces la o instalare funcţională a acestor tehnologii. Chiar şi în cazul în care nu veţi fi nevoiţi să instalaţi MySQL, Apache şi PHP, ar trebui să parcurgeţi totuşi aceste capitole pentru a fi siguri că înţelegeţi elementele esenţiale.
- Partea a II-a, *Structura limbajului PHP*, este dedicată iniţierii dumneavoastră în bazele limbajului PHP, incluzând elemente structurale precum vectorii şi obiectele. Exemplele vă vor ajuta să începeţi să scrieţi cod, să încărcaţi fişierele pe serverul de testare şi să testaţi rezultatele.
- Partea a III-a, *La lucru cu codul*, este formată din lecţii care acoperă probleme legate de dezvoltarea de aplicaţii de nivel intermediar, inclusiv lucrul cu formularele şi fişierele, restricţionarea accesului şi completarea altor mici proiecte destinate prezentării unor concepte specifice.
- Partea a IV-a, *Integrarea PHP cu MySQL*, conţine lecţii destinate lucrului cu bazele de date în general, cum ar fi normalizarea bazelor de date, precum şi folosirea limbajului PHP pentru conectarea la baze de date MySQL şi utilizarea înregistrărilor din acestea. Tot aici sunt discutate şi noţiunile primare de SQL, care includ şi funcţii specifice de MySQL şi alte informaţii.
- Partea a V-a, *Proiecte fundamentale*, este formată din lecţii dedicate îndeplinirii anumitor obiective folosind PHP şi MySQL, integrând cunoştinţele acumulate până în acest punct. Proiectele includ printre altele o agendă personală cu adrese, un forum de discuţii şi un magazin virtual cu funcţii de bază.
- Partea a VI-a, *Administrare şi optimizare*, este dedicată administrării şi optimizării serverului Apache şi a bazei de date MySQL. De asemenea, include informaţii despre găzduirea virtuală şi configurarea unui server Web securizat.
- Partea a VII-a, *Gânduri spre viitor*, conţine informaţii despre versiunile noi aflate în pregătire, şi anume PHP 5.0 şi MySQL 4.1.

În cazul în care consideraţi că sunteţi deja familiarizat cu un anumit subiect, puteţi sări direct la următoarea lecţie. Totuşi, în anumite circumstanţe, lecţiile conţin referiri la concepte dezvoltate în capitolele anterioare, deci există posibilitatea de a fi nevoiţi să reveniţi şi să răsfoiţi o lecţie peste care aţi sărit anterior astfel încât mediul dumneavoastră de dezvoltare să rămână asemănător celui prezentat în carte.

La sfârşitul fiecărui capitol, câteva întrebări scurte vor testa nivelul de asimilare al informaţiilor furnizate. Activităţile suplimentare vă oferă posibilitatea utilizării cunoştinţelor acumulate în respectiva lecţie şi vă ghidează în vederea folosirii acestora în capitolele următoare.

Convenţii utilizate în carte

În această carte se utilizează diferite stiluri pentru a vă ajuta să faceţi diferenţa între secţiunile de cod de programare şi limbajul obişnuit şi, de asemenea, pentru a vă ajuta să identificaţi conceptele importante. Pe parcursul lecţiilor, codul de programare, comenzile şi textul pe care trebuie să îl tastaţi sau să îl citiţi de pe ecran va apărea formatat intr-un stil de tip `consolă de computer`. Termenii noi apar formataţi cu litere *cursive* în cadrul paragrafului unde sunt definiţi. De asemenea, blocurile de informaţii speciale sunt acompaniate de pictograme:

Pictograma „Apropo" vă semnalează o informaţie interesantă relativă la subiectul aflat în discuţie.

Apropo

Pictograma „Ştiaţi că..." vă punctează locul unde vi se oferă sfaturi sau vi se dezvăluie metode mai simple de a efectua o sarcină.

Ştiaţi că...

Pictograma „Atenţie!" vă precizează diferite potenţiale capcane sau pericole şi vă explică metode de a le evita.

Atenţie!

PARTEA I

Elemente fundamentale

Capitolul 1. Instalarea şi configurarea serverului MySQL 7

Capitolul 2. Instalarea şi configurarea serverului Apache 27

Capitolul 3. Instalarea şi configurarea pachetului PHP 51

CAPITOLUL 1

Instalarea şi configurarea serverului MySQL

Bine aţi venit la primul capitol din lucrarea *Învaţă singur PHP, MySQL şi Apache — Toate într-o singură carte*. Acesta este primul dintre cele trei capitole dedicate instalărilor, în care veţi învăţa cum să configuraţi un mediu de dezvoltare. Vom aborda pentru început instalarea MySQL, datorită faptului ca instalarea PHP se face mult mai uşor atunci când MySQL este deja instalat şi configurat.

În acest capitol, veţi învăţa:

- cum să instalaţi MySQL;
- considerente de bază privind securitatea rulării serverului MySQL;
- cum să lucraţi cu sistemul de utilizatori şi privilegii al serverului MySQL.

Versiuni curente şi viitoare de MySQL

Instrucţiunile de instalare din acest capitol sunt specifice pentru versiunea 4.0.15, care este versiunea curentă. Acest număr de versiune poate fi interpretat ca fiind pachetul cu numărul 15, pus la dispoziţie în cadrul versiunii 4.0. MySQL AB, compania responsabilă de crearea şi distribuţia de MySQL, foloseşte numărul de pachet pentru versiuni conţinând îmbunătăţiri ale securităţii sau rezolvări de erori de programare (bugs). Pachetele nu sunt puse la dispoziţia publicului după un program bine determinat şi prestabilit; când sunt adăugate îmbunătăţiri sau rezolvări de probleme şi când acestea sunt testate corespunzător, MySQL AB publică o nouă versiune, cu un nou număr de pachet.

Există posibilitatea ca în momentul în care citiţi această carte numărul de versiune să se fi schimbat în 4.0.16 sau chiar mai mult. În acest caz ar trebui să consultaţi lista cu schimbări disponibilă la adresa Web `http://www.mysql.com/doc/en/News-4.0.x.html` şi să urmăriţi eventualele diferenţe în ceea ce priveşte procesul de instalare şi configurare, care constituie subiectul acestui capitol.

Deşi este puţin probabil ca instrucţiunile să se schimbe substanţial de la o versiune la alta, ar trebui să vă formaţi obiceiul de a consulta mereu lista de schimbări pentru aplicaţiile pe care le instalaţi şi administraţi. Dacă versiunea evoluează până la momentul în care citiţi această carte, dar nu sunt înregistrate

schimbări legate de procesul de instalare, amintiţi-vă pur şi simplu să substituiţi noul număr de versiune oriunde apare acesta în instrucţiunile de instalare sau în figurile care le însoţesc.

Cum găsiți kitul de instalare MySQL

MySQL AB, compania care dezvoltă şi distribuie serverul de baze de date MySQL, îl pune la dispoziţia publicului prin intermediul site-ului Web `http://www.mysql.com/`. Distribuţiile binare pentru toate platformele, precum şi pachetele RPM şi fişierele cu surse pentru platforme Linux/Unix pot fi găsite pe acest site. De asemenea, puteţi cumpăra pachete comerciale — care conţin aplicaţia şi o versiune tipărită a manualului extrem de cuprinzător al MySQL — direct de la magazinul virtual MySQL AB, contra unei sume rezonabile.

Apropo

Pentru a vedea instrucțiunile de instalare MySQL, utile în situația în care doriți să folosiți CD-ROM-ul ataşat cărții, vă rugăm să consultați anexa A, intitulată *Instalarea aplicațiilor MySQL, Apache şi PHP aflate pe CD-ROM-ul ataşat cărții.*

Instalarea MySQL în cadrul sistemului de operare Linux/Unix

Procesul de instalare MySQL pe Linux/Unix este foarte clar, fie că folosiți pachete RPM, fie că instalaţi fişierele binare. Pentru o instalare minimă folosind pachete RPM aveți nevoie de două fişiere:

- `MySQL-server-VERSION.i386.rpm` — serverul MySQL,
- `MySQL-client-VERSION.i386.rpm` — bibliotecile standard MySQL.

Pentru a efectua o instalare minimală folosind pachete RPM, tastaţi următoarea comandă în terminalul dumneavoastră:

```
#> rpm -i MySQL-server-VERSION.i386.rpm MySQL-client-VERSION.i386.rpm
```

Apropo

Înlocuiți `VERSION` în numele fişierului cu numărul versiunii pe care ați descărcat-o. De exemplu, versiunea curentă a pachetului RPM server MySQL 4.0 se numeşte `MySQL-server-4.0.15-0.i386.rpm`, iar pachetul RPM conținând bibliotecile client standard se numeşte `MySQL-client-4.0.15-0.i386.rpm`.

O altă metodă confortabilă este să instalaţi MySQL folosind o distribuţie binară. Această metodă necesită folosirea utilitarelor `gunzip` şi `tar` pentru a decomprima şi despacheta distribuţia, de asemenea, fiind necesar să aveţi dreptul de a crea grupuri şi utilizatori în cadrul sistemului. Prima serie de comenzi vă permite crearea unui grup şi a unui utilizator şi despachetarea distribuţiei:

Apropo

Înlocuiţi `VERSION-OS` în numele fişierului cu numărul versiunii pe care aţi descărcat-o. De exemplu, versiunea curentă a distribuţiei binare MySQL 4.0 Linux/i386 se numeşte `mysql-max-4.0.15-pc-linux-i686.tar.gz`.

```
#> groupadd mysql
#> useradd -g mysql mysql
#> cd/usr/local
#> gunzip < /path/to/mysql-VERSION-OS.tar.gz | tar xvf -
```

Următoarele instrucţiuni vă permit să creaţi o legătură cu un nume mai scurt:

```
#> ln - s mysql-VERSION-OS mysql
#> cd mysql
```

O dată despachetată distribuţia, fişierele `README` şi `INSTALL` vă vor ghida pe parcursul restului procesului de instalare pentru versiunea pe care aţi ales-o. În general va fi folosită următoarea serie de comenzi:

```
#> scripts/mysql_install_db
#> chown -R root /usr/local/mysql
#> chown -R mysql /usr/locql/mysql/data
#> chown -R mysql /usr/local/mysql/
#> chown -R root /usr/local/mysql/bin
```

Sunteţi gata să porniţi serverul MySQL, aşa că puteţi sări direct la secţiunea intitulată *Considerente fundamentale de securitate*, de la pagina 16. Dacă aţi întâmpinat vreo eroare în cadrul procesului de instalare verificaţi secţiunea intitulată *Depanarea procesului de instalare*, aflată tot la pagina 16.

Instalarea MySQL în cadrul sistemului de operare Windows

Procesul de instalare MySQL este de asemenea foarte simplu şi pe sistemul de operare Windows — dezvoltatorii de la MySQL AB au pus la dispoziţia publicului toate fişierele necesare, alături de un program de instalare într-o simplă arhivă zip! Descărcaţi fişierul zip, extrageţi conţinutul acestuia într-un fişier temporar şi rulaţi aplicaţia `setup.exe`. După instalarea serverului şi bibliotecilor client, puteţi porni serverul MySQL.

Următorii paşi detaliează instalarea distribuţiei `MySQL 4.0.x` pe sistemul de operare Windows, folosind fişierele de instalare descărcate de la MySQL AB. Secvenţa de instalare arată aproximativ identic indiferent de versiunea sistemului

de operare, Windows 98, Windows NT, Windows 2000 sau Windows XP pe care îl folosiţi ca mediu de testare şi dezvoltare. Mulţi utilizatori îşi instalează serverul MySQL pe calculatorul personal, care rulează sistemul de operare Windows, doar pentru a-şi face o impresie despre lucrul cu bazele de date înainte de a implementa MySQL într-un mediu de producţie.

Apropo

Dacă dispuneţi de instrumentele şi aptitudinile necesare pentru a vă compila propriile fişiere binare, codul sursă în format Cygwin este de asemenea disponibil pentru a fi descărcat de la MySQL AB. Urmăriţi instrucţiunile incluse in distribuţia codurilor sursă pentru a vă compila propriile fişiere executabile.

Trecem acum direct la secvenţa de instalare. Presupunând că aţi descărcat deja fişierele de instalare pentru platforma Windows de pe situl MySQL AB, urmaţi aceşti paşi:

1. Extrageţi conţinutul arhivei zip într-un director temporar, identificaţi fişierul `setup.exe` şi daţi clic dublu pentru a iniţializa procedura de instalare. Veţi vedea ecranul de întâmpinare al utilitarului de asistenţă pentru instalare aşa cum vă este prezentat în figura 1.1. Apăsaţi butonul Next (următorul) pentru a continua.

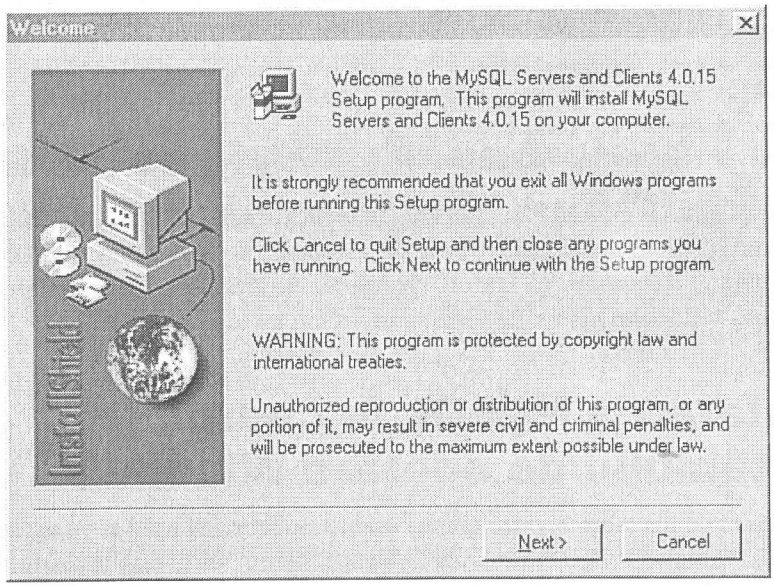


FIGURA 1.1. Ecranul de întâmpinare al utilitarului de asistenţă pentru instalare al serverului MySQL.

2. Al doilea ecran conţine informaţii foarte importante despre directorul de instalare (consultaţi figura 1.2). Directorul implicit de instalare este `C:\mysql`. Dacă intenţionaţi să schimbaţi directorul implicit şi să instalaţi serverul intr-o locaţie aleasă de dumneavoastră, acest ecran vă pune la dispoziţie informaţii despre configurările pe care va trebui să le efectuaţi manual pentru a realiza această modificare. Informaţiile din acest ecran sunt deosebit de importante şi pentru utilizatorii de Windows NT care

vor dori să pornească serverul MySQL ca serviciu. Citiţi cu atenţie informaţiile şi notaţi-vă tot ceea ce vi se pare relevant pentru situaţia dumneavoastră, apoi apăsaţi butonul Next pentru a continua.

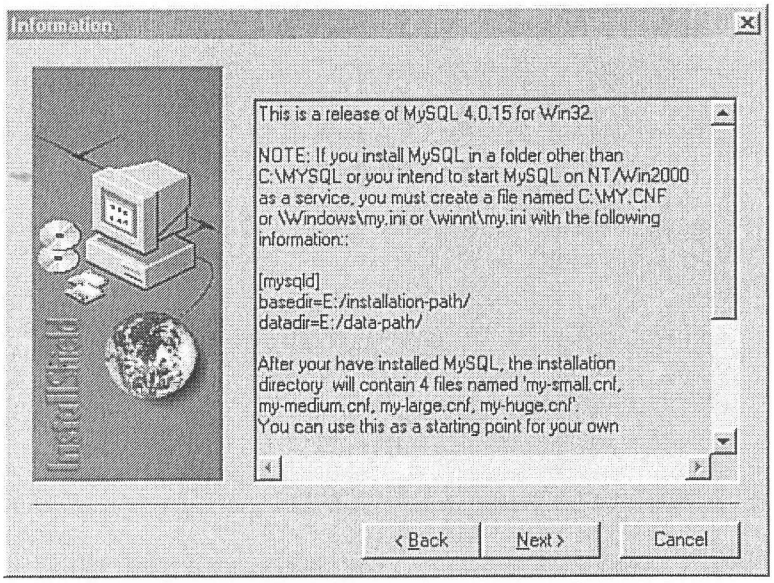


FIGURA 1.2. Pasul 2 in procesul de instalare a serverului MySQL. Notaţi-vă orice informaţie relevantă înainte de a continua.

3. Al treilea ecran din procesul de instalare vă invită să alegeţi directorul de instalare (aşa cum vi se arată în figura 1.3). Dacă doriţi să folosiţi directorul implicită, sugerat de utilitarul de asistenţă, daţi clic pe butonul Next pentru a continua. Dacă doriţi să îl modificaţi, selectaţi butonul Browse şi navigaţi până la directorul dorit, apoi daţi clic pe butonul Next pentru a continua.

4. Al patrulea ecran vă invită să alegeţi metoda de instalare — Typical (instalare tipică), Compact (instalare compactă) sau Custom (instalare personalizată), ca în figura 1.4. Opţiunea de instalare personalizată (Custom) vă permite să selectaţi, pentru a fi instalate, elemente suplimentare, ca de exemplu documentaţiile sau fişierele de asistenţă. Selectaţi metoda de instalare tipică (opţiunea Typical) şi apăsaţi butonul Next pentru a continua.

5. Procesul de instalare va prelua acum controlul şi va instala fişierele in directoarele adecvate. La terminarea acestui proces veţi fi informat printr-un ecran de confirmare, similar celui din figura 1.5. Apăsaţi butonul Finish pentru a încheia procesul de instalare.

După acest tip de instalare, folosind fişierele descărcate de la MySQL AB, nu veţi găsi scurtături în meniul Start din Windows, aşa că veţi fi nevoiţi să porniţi aplicaţia manual. Dacă navigaţi până la directorul aplicaţiei MySQL (de obicei `C:\mysql\bin` — în cazul în care nu aţi modificat directorul de instalare), veţi descoperi aici numeroase aplicaţii gata de a fi rulate (ca în figura 1.6).

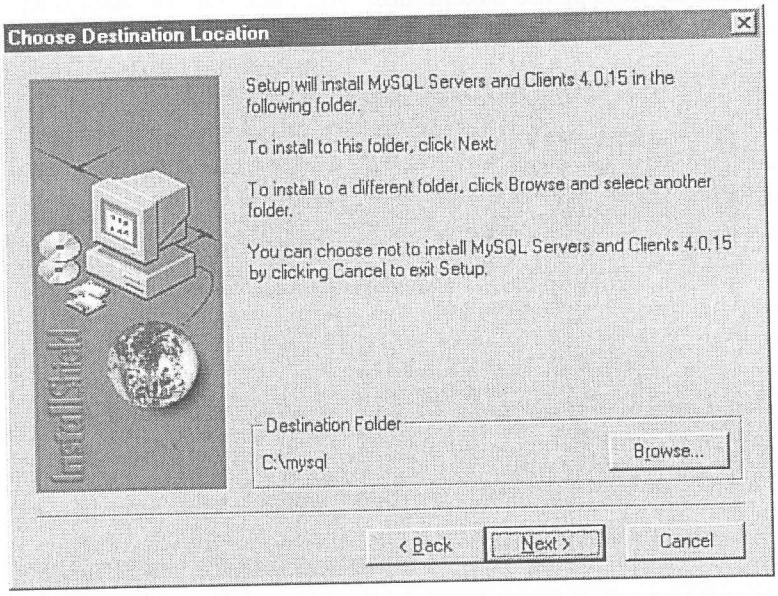


FIGURA 1.3. Pasul 3 din procesul de instalare MySQL. Selectați locul instalării.

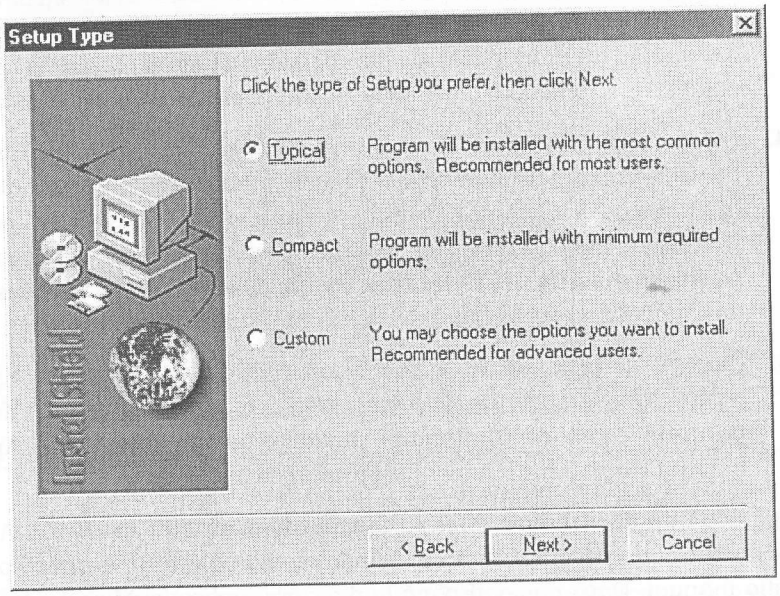


FIGURA 1.4. Pasul 4 din procesul de instalare MySQL. Selectați o metodă de instalare.

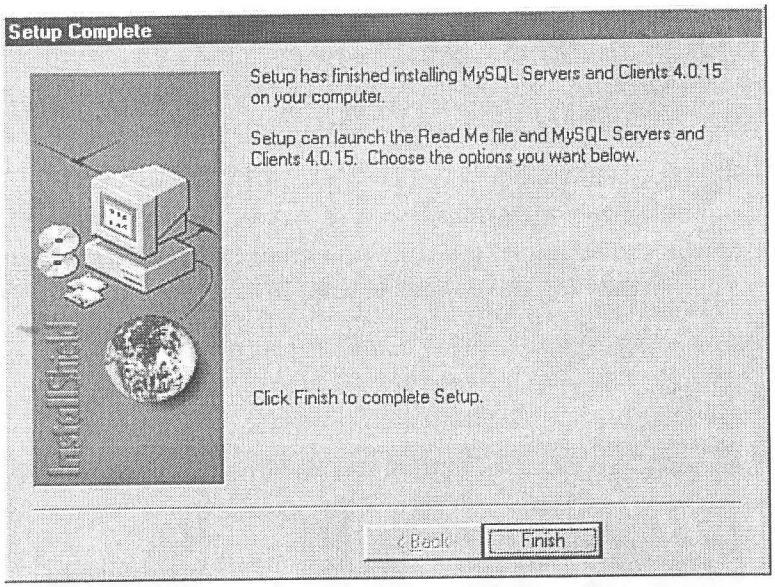


FIGURA 1.5. MySQL a fost instalat.

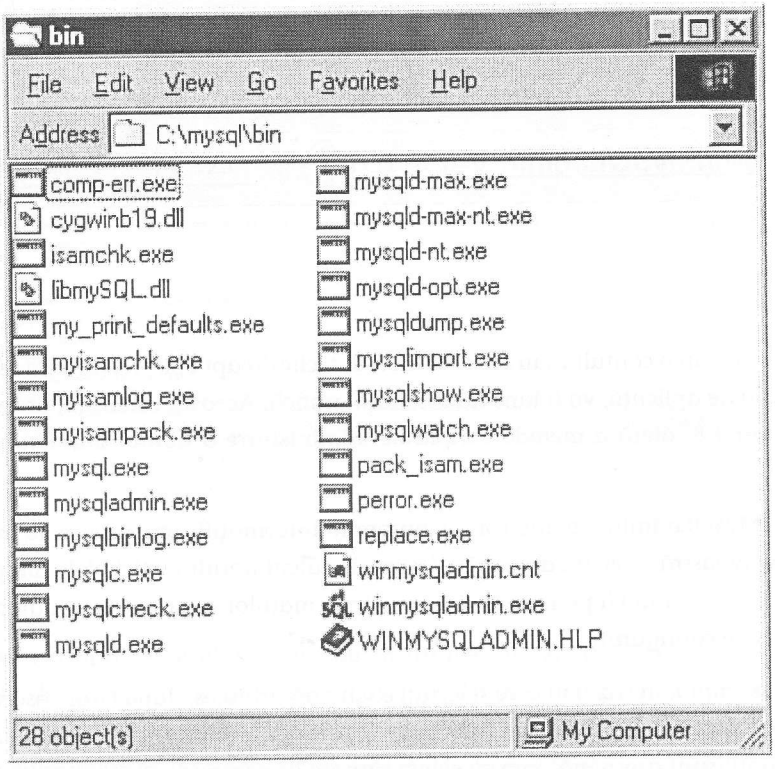


FIGURA 1.6. Afişarea directorului cu aplicaţiile MySQL.

Aplicaţia `winmysqladmin.exe` este un prieten de neînlocuit al tuturor utilizatorilor de Windows care sunt la început de drum în utilizarea MySQL. În cazul în care daţi clic dublu pe acest fişier, acesta va iniţializa serverul MySQL şi va plasa în bara de aplicaţii o pictogramă asemănătoare unui semafor.

La prima iniţializare `WinMySQLadmin` vi se va cere introducerea unui nume de utilizator şi a unei parole, aşa cum vi se arată în figura 1.7. Aplicaţia va crea contul iniţial MySQL pe sistemul de operare Windows.

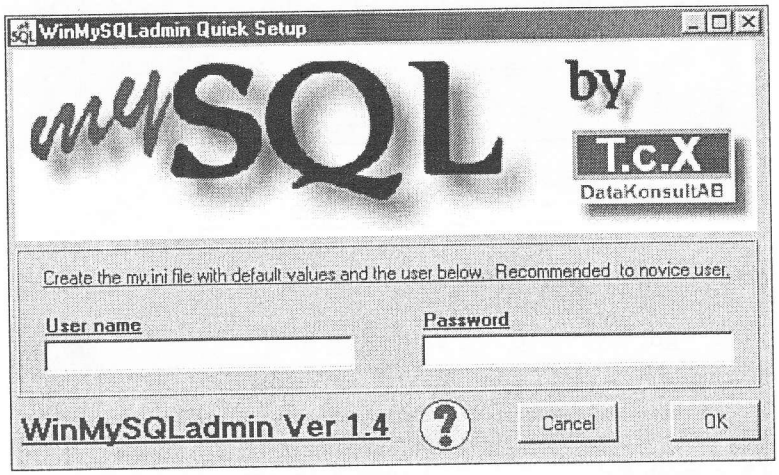


FIGURA 1.7. Crearea contului iniţial MySQL.

După crearea contului sau ori de câte ori daţi clic dreapta pe pictograma semafor din bara de aplicaţii, va fi lansată interfaţa grafică. Această interfaţă, prezentată în figura 1.8, oferă o metodă simplă de administrare a noului dumneavoastră server.

WinMySQLadmin va interpreta automat informaţiile cu privire la mediul dumneavoastră, precum adresa IP şi numele calculatorului. Etichetele din partea de sus a ecranului vă permit vizualizarea informaţiilor de sistem şi editarea unor opţiuni de configurare.

De exemplu, în cazul în care selectaţi eticheta Variables, după cum observaţi în figura 1.9, puteţi să vizualizaţi informaţii legate de configuraţia serverului. Aceste informaţii sunt asemănătoare cu cele returnate de comanda `SHOW VARIABLES` din MySQL.

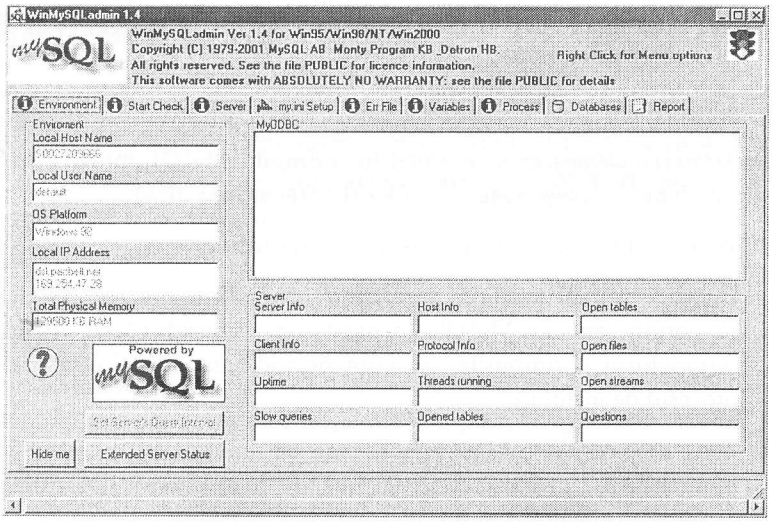


FIGURA 1.8. WinMySQL-admin iniţiat şi gata de acţiune.

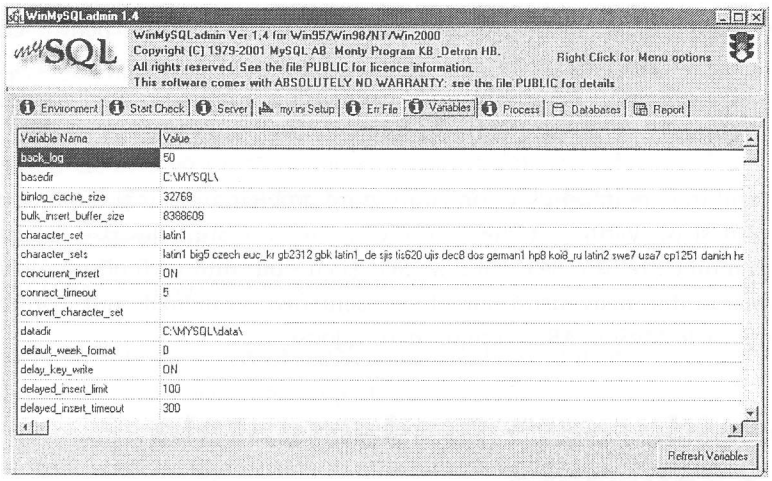


FIGURA 1.9. Informaţiile despre configuraţia serverului.

Pentru a opri serverul MySQL sau instrumentul WinMySQLadmin, daţi din nou clic dreapta pe pictograma semafor din bara de aplicaţii, după care selectaţi opţiunile dorite (Stop sau Start). Atâta vreme cât serverul MySQL este în funcţiune, puteţi rula aplicaţii adiţionale prin intermediul unei ferestre consolă, cum este cazul ferestrei MySQL Monitor.

Depanarea procesului de instalare

În caz că întâmpinaţi probleme pe parcursul procesului de instalare a serverului MySQL, primul loc unde ar trebui să căutaţi o rezolvare este capitolul *Problems and Common Errors (Probleme şi erori frecvente)* din manualul MySQL, manual disponibil la adresa Web `http://www.mysql.com/doc/P/r/Problems.html`.

Următoarele sunt considerate probleme des întâlnite:

- Pe platforma Linux/Unix, permisiuni configurate incorect nu vă permit să porniţi serviciul MySQL. În acest caz, asiguraţi-vă că aţi schimbat proprietarul şi grupurile aşa cum prevăd instrucţiunile referitoare la procesul de instalare.
- În cazul în care primiţi mesajul `Access denied` (Accesul nu este permis) în momentul în care încercaţi să vă conectaţi la serverul MySQL, asiguraţi-vă că folosiţi utilizatorul şi parola corectă.
- În cazul în care mesajul primit este `Can't connect to server` (Nu vă puteţi conecta la server), asiguraţi-vă ca serviciul MySQL a fost pornit şi că rulează fără probleme.
- Atunci când definiţi tabele, dacă specificaţi lungimi pentru câmpuri ale căror tipuri nu cer specificarea unei lungimi, tabela nu va fi creată. De exemplu, nu ar trebui să specificaţi o lungime atunci când definiţi câmpuri de tip `Text` (spre deosebire de `CHAR` sau `VARCHAR`).

În cazul în care întâmpinaţi în continuare probleme, chiar şi după ce aţi citit manualul şi aţi verificat toate erorile comune care sunt descrise, trimiteţi un mesaj e-mail către lista de discuţii MySQL şi cu siguranţă va fi găsită o rezolvare la problema dumneavoastră. Puteţi, de asemenea, să achiziţionaţi contracte de asistenţă de la MySQL AB, la preţuri avantajoase.

Considerente fundamentale de securitate

Indiferent dacă rulaţi serverul MySQL pe o platformă Windows sau sub Linux/Unix şi indiferent dacă administraţi propriul server sau folosiţi un sistem oferit de furnizorul dumneavoastră de servicii Internet, trebuie să înţelegeţi considerentele fundamentale de securitate. Dacă accesaţi serverul MySQL prin furnizorul dumneavoastră de servicii Internet există câteva aspecte ale securităţii serverului pe care dumneavoastră ca utilizator `non-root` trebuie să puteţi să le modificaţi sau să le ocoliţi. Din păcate mulţi din furnizorii de servicii Internet nu acordă nici o importanţă securităţii serverelor, lăsându-şi clienţii expuşi la riscuri, a căror existenţă, de multe ori, aceştia nici nu o cunosc.

Pornirea serverului MySQL

Securizarea serverului MySQL începe chiar cu procedura de iniţializare a acestuia. Dacă nu sunteţi administratorul serverului, nu veţi putea să schimbaţi această procedură, dar puteţi, fără îndoială, să o analizaţi şi să sesizaţi furnizorul dumneavoastră de servicii Internet cu privire la vulnerabilităţile identificate de dumneavoastră.

Dacă instalarea MySQL s-a realizat pe o platformă Linux/Unix, prima dumneavoastră grijă ar trebui să fie proprietarul serviciului MySQL — în nici un caz acesta nu ar trebui să fie utilizatorul `root`. Rularea serviciului sub un utilizator non-`root`, ca de exemplu `mysql` sau `database`, limitează posibilităţile indivizilor răuvoitori de a dobândi acces la server şi de a suprascrie fişierele.

Puteţi verifica proprietarul procesului folosind comanda `ps` (acronim pentru process status — statutul proceselor) pe sistemul dumneavoastră Linux/Unix. Următoarele linii vă arată că serverul MySQL rulează sub un utilizator `non-root` (observaţi primul cuvânt din cea de-a doua linie):

```
#> ps auxw | grep mysqld
mysql 153  0.0  0.6 12068 2624 ? S Nov16  0:00 /usr/local/bin/mysql/bin/mysqld
--defaults-extra-file=/usr/local/bin/mysql/data/my.cnf
--basedir=/usr/local/bin/mysql --datadir=/usr/local/bin/mysql/data
--user=mysql --pid-file=/usr/local/bin/mysql/data/mike.pid --skip-locking
```

Următoarele linii vă arată că serverul MySQL rulează sub un utilizator `root` (aşa cum vă arată primul cuvânt din cea de-a doua linie):

```
#> ps auxw | grep mysqld
root 21107  0.0  1.1 11176  1444  ?  S  Nov 27 0:00 /usr/local/mysql/bin/mysqld
--basedir=/usr/local/mysql --datadir=/usr/local/mysql/data --skip-locking
```

Dacă remarcaţi că serverul MySQL rulează sub un utilizator `root` pe sistemul dumneavoastră, contactaţi de urgenţă furnizorul dumneavoastră de servicii Internet şi reclamaţi acest lucru. Dacă sunteţi administratorul sistemului, trebuie să iniţializaţi serviciul MySQL sub un utilizator non-`root` sau să specificaţi utilizatorul în comanda de iniţializare:

```
mysqld --user=non_root_user_name
```

De exemplu, dacă doriţi să rulaţi MySQL sub utilizatorul `mysql`, folosiţi comanda:

```
mysqld --user=mysql
```

Totuşi, metoda recomandată de iniţializare a serviciului este rularea scriptului `safe_mysqld` localizat în directorul `bin` al instalării aflate pe sistemul dumneavoastră:

```
#> /usr/local/bin/mysql/bin/safe_mysqld &
```

Securizarea conexiunii la MySQL

Vă puteţi conecta la programul MySQL monitor sau la altă aplicaţie în mai multe moduri, fiecare fiind supus unor riscuri de securitate. Dacă instalarea serverului MySQL se află pe calculatorul dumneavoastră, aveţi mai puţine griji în comparaţie cu utilizatorii care trebuie să se conecteze la server prin reţea.

Dacă serverul MySQL este instalat local pe calculatorul dumneavoastră, cea mai mare grijă a dumneavoastră este de a nu lăsa calculatorul nesupravegheat, şi cu instrumente de administrare, MySQL monitor sau MySQL GUI, pornite şi la vedere. Într-o astfel de situaţie oricine trece prin zonă poate să şteargă date, să introducă date false sau să oprească serverul. Dacă trebuie să lăsaţi calculatorul nesupravegheat în zone publice, folosiţi aplicaţii de tipul screen-saver sau mecanisme de blocare cu parolă a ecranului.

Dacă serverul MySQL este instalat pe un server din afara reţelei dumneavoastră, securizarea conexiunii ar trebui să fie prima prioritate pentru dumneavoastră. Ca orice altă transmisiune de date prin Internet, şi aceasta ar putea fi interceptată. Dacă transmisiunea este necriptată, persoana care o interceptează o poate reconstitui şi folosi. Să presupunem că transmisia necriptată conţine informaţia de autentificare pentru serverul MySQL — un individ răuvoitor are acum acces la baza dumneavoastră de date şi poate întreprinde diferite acţiuni folosind acreditările dumneavoastră.

O metodă sigură de a preveni astfel de întâmplări este să vă conectaţi la serverul dumneavoastră MySQL folosind o conexiune securizată. În loc să vă conectaţi la calculatorul aflat la distanţă folosind clasicul Telnet, conectaţi-vă prin intermediul SSH. SSH arată şi se comportă la fel ca Telnet, dar toate transmisiunile de la şi către calculatorul aflat la distanţă sunt criptate. De asemenea dacă folosiţi o interfaţă de administrare bazată pe tehnologie Web, cum este cazul phpMyAdmin (pentru mai multe informaţii accesaţi `http://phpmyadmin.sourceforge.net`) sau orice alte instrumente puse la dispoziţie de furnizorul dumneavoastră de servicii Internet, aveţi grijă sa le accesaţi prin intermediul unei conexiuni HTTP securizate.

În următoarea secţiune veţi învăţa despre sistemul de privilegii al MySQL, un instrument foarte util în procesul de securizare a bazei de date.

Introducere în sistemul de privilegii al MySQL

Sistemul de privilegii al MySQL este întotdeauna pornit. Prima dată când încercaţi să vă conectaţi şi ulterior pentru orice acţiune întreprinsă, serverul MySQL verifică următoarele trei aspecte:

- de unde accesaţi serverul (host-ul dumneavoastră);
- cine sunteţi sau pretindeţi că aţi fi (utilizatorul şi parola);
- ce operaţii aveţi dreptul să întreprindeţi (privilegiile dumneavoastră de comandă).

Toate aceste informaţii sunt stocate într-o bază de date intitulată `mysql`, care se creează automat în momentul instalării. În această bază de date întâlniţi mai multe tabele:

- `columns_priv` — conţine informaţiile referitoare la privilegiile unui anumit utilizator pentru anumite câmpuri dintr-o tabelă;
- `db` — conţine informaţiile referitoare la permisiunile pentru toate bazele de date de pe server;
- `func` — conţine funcţiile create de utilizator;
- `host` — conţine lista tuturor staţiilor (host-urilor) de unde sistemul poate accepta conexiuni;
- `tables_priv` — conţine informaţiile cu privire la privilegiile unui anumit utilizator pe tabele specifice dintr-o bază de date;
- `user` — conţine informaţiile referitoare la comenzile pe care un anumit utilizator are dreptul să le execute sau nu.

Aceste tabele vor căpăta o mai mare importanţă ulterior în acest capitol pe măsură ce veţi adăuga exemple de utilizatori de MySQL. Până atunci trebuie doar să reţineţi că aceste tabele există şi că ele trebuie să conţină datele necesare pentru ca un utilizator să poată întreprinde o anumită acţiune.

Procesul de autentificare în doi paşi

Aşa cum aţi aflat deja, serverul MySQL verifică trei lucruri în cadrul procesului de autentificare. Acţiunile asociate cu aceste trei lucruri sunt efectuate în doi paşi:

1. Serverul MySQL verifică staţia (`host`-ul) de unde vă conectaţi, precum şi perechea utilizator — parolă pe care aţi furnizat-o. Dacă staţia dumneavoastră este autorizată să se conecteze, parola corespunde utilizatorului şi utilizatorul este cel asociat staţiei respective, serverul MySQL trece la pasul următor.
2. Pentru orice comandă SQL pe care încercaţi să o rulaţi, serverul MySQL verifică dacă sunteţi autorizat să executaţi respectiva comandă asupra bazei de date, tabelei sau câmpului vizat.

Dacă sunt întâlnite erori la pasul 1, veţi vedea un mesaj care descrie respectivele erori şi nu veţi putea trece la pasul 2. De exemplu: să presupunem că vă conectaţi la serverul MySQL folosind numele de utilizator `joe` şi parola `abc123` şi că doriţi să accesaţi o bază de date intitulată `myDB`. Veţi primi un mesaj de eroare dacă oricare dintre aceste variabile de conectare sunt incorecte, din cauza unuia dintre următoarele motive:

- parola este incorectă;
- utilizatorul `joe` nu există;
- utilizatorul `joe` nu are dreptul de a se conecta de pe staţia `localhost`;
- utilizatorul `joe` are dreptul de a se conecta de pe staţia `localhost`, dar nu are dreptul să utilizeze baza de date `myDB`.

Există şi posibilitatea să vedeţi o eroare precum cea de mai jos:

```
#> /usr/local/mysql/bin/mysql -h localhost -u joe -pabc123 test
Error 1045: Access denied for user: 'joe@localhost' (Using password: YES)
```

Dacă utilizatorului `joe` cu parola `abc123` îi este permisă conectarea de la staţia `localhost`, precum şi utilizarea bazei de date `myDB`, MySQL va verifica la pasul 2 al procesului de conectare acţiunile pe care joe are dreptul să le întreprindă. În scop demonstrativ, să presupunem că `joe` are dreptul de a selecta (`SELECT`) datele, dar nu are dreptul de a introduce (`INSERT`) date noi. Secvenţa de evenimente şi erori ar arăta precum în liniile de cod de mai jos:

```
#> /usr/local/mysql/bin/mysql -h localhost -u joe -pabc123 test
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 61198 to server version: 4.0.15-log
Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> select * from test_table;
+----+------------+
| id | test_field |
+----+------------+
+----+------------+
|  1 | blah       |
|  2 | blah blah  |
+----+------------+
2 rows in set (0.0 sec)

mysql> insert into test_table values ('', 'my text');
Error 1044: Access denied for user: 'joe@localhost' (Using password: YES)
```

Permisiunile bazate pe acţiuni sunt foarte des întâlnite în aplicaţiile cu mai multe nivele de administrare. De exemplu, dacă aţi creat o aplicaţie care conţine informaţii financiare personale, aţi putea alege să oferiţi personalului normal doar posibilitatea de a selecta (`SELECT`) date, dar de asemenea aţi putea oferi personalului de conducere, care beneficiază de autorizările necesare, posibilitatea şi de a insera (`INSERT`) sau şterge (`DELETE`) date.

Lucrul cu privilegiile acordate utilizatorilor

În cele mai multe cazuri, atunci când accesaţi serverul MySQL prin intermediul furnizorul dumneavoastră de servicii Internet, veţi avea la dispoziţie doar un utilizator şi o singură bază de date. Iniţial, acel utilizator va avea acces la toate tabelele din acea bază de date şi va fi acreditat să efectueze orice acţiune asupra ei. În acest caz, responsabilitatea de a crea prin programare o aplicaţie sigură vă aparţine dumneavoastră ca dezvoltator.

Dacă sunteţi administratorul serverului propriu de baze de date sau sunteţi abilitat să adăugaţi oricâte baze de date şi utilizatori doriţi, precum şi să modificaţi drepturile şi privilegiile de acces ale utilizatorilor dumneavoastră, atunci următoarele secţiuni vă vor oferi informaţii foarte importante despre aceşti paşi.

Adăugarea utilizatorilor

Administrarea serverului dumneavoastră cu ajutorul unui instrument dezvoltat de o terţă parte v-ar putea oferi metode foarte simple de adăugare a utilizatorilor, cu ajutorul unor procese de tipul utilitarelor de asistenţă sau a unei interfeţe grafice specifice. Totuşi, adăugarea de utilizatori cu ajutorul utilitarului MySQL monitor nu este deloc dificilă , mai ales dacă sunteţi deja familiarizat cu conceptele de securitate ale serverului MySQL care v-au fost prezentate anterior.

Cea mai simplă metodă de a adăuga un utilizator nou este comanda `GRANT`. Conectându-vă la serverul MySQL ca utilizator `root` puteţi să tastaţi această comandă care vă permite adăugarea de noi utilizatori. Cealaltă metodă este utilizarea comenzii `INSERT` asupra tuturor tabelelor relevante din baza de date `mysql`, caz în care este necesară o cunoaştere temeinică a câmpurilor şi tabelelor în care sunt stocate permisiunile. Metoda comenzilor `INSERT` succesive produce acelaşi rezultat dar este puţin mai laborioasă şi mai complicată decât comanda clasică `GRANT`.

Sintaxa simplă a comenzii `GRANT` este următoarea:

```
GRANT privilegii
ON nume_baza_de_date.nume_tabela
TO nume_utilizator@host
IDENTIFIED BY "parola";
```

Privilegiile pe care le puteţi oferi utilizatorilor prin intermediul comenzii `GRANT` sunt următoarele:

- `ALL` — conferă utilizatorului toate privilegiile de mai jos;
- `ALTER` — utilizatorii pot modifica tabele, coloane şi indexuri;
- `CREATE` — utilizatorii pot crea baze de date, tabele şi indexuri noi;
- `DELETE` — utilizatorii pot şterge înregistrări din tabele;
- `DROP` — utilizatorii pot muta tabele şi baze de date;
- `FILE` — utilizatorii pot citi şi scrie fişiere; procedură folosită la importul şi exportul (`dump`) datelor din baza de date;
- `INDEX` — utilizatorii pot adăuga sau şterge indexuri;
- `INSERT` — utilizatorii pot introduce date noi în tabele;
- `PROCESS` — utilizatorii pot vizualiza şi interacţiona cu procesele de sistem, oprindu-le, resetându-le sau pornindu-le; ca măsură de securitate, doar administratorii şi utilizatorii de mare încredere ar trebui să poată beneficia de această facilitate;

- REFERENCES — opţiunea nu este utilizată în prezent de către serverul MySQL; există totuşi în tabelul user o coloană rezervată pentru privilegii privind referinţele;
- RELOAD — utilizatorii pot comanda resetarea serverului (FLUSH); opţiunea este recomandată numai administratorilor şi utilizatorilor de mare încredere;
- SELECT — utilizatorii pot selecta şi vizualiza înregistrări din baza de date;
- SHUTDOWN — utilizatorii pot opri serverul MySQL; opţiune recomandată numai administratorilor şi utilizatorilor de mare încredere;
- UPDATE — utilizatorii pot modifica şi actualiza datele din tabelele bazei de date;
- USAGE — utilizatorii se pot conecta la serverul MySQL, dar nu au nici un alt fel de privilegiu.

Dacă, de exemplu, doriţi să creaţi un utilizator numit john căruia să îi fie asociată parola 99hjc şi care să aibă privilegiile de a selecta (SELECT) şi introduce (INSERT) date în toate tabelele din baza de date numită myDB şi doriţi, de asemenea, ca acest utilizator să se poată conecta de la orice staţie, ar trebui să scrieţi o comandă ca cea de mai jos:

```
GRANT SELECT, INSERT
ON myDB.*
TO john@"%"
IDENTIFIED BY "99hjc";
```

Observaţi utilizarea celor două caractere generale „joker" * şi %. Aceste caractere generale sunt utilizate pentru a înlocui toate valorile posibile. În acest exemplu * înlocuieşte întreaga listă de tabele din baza de date myDB, iar % înlocuieşte lista tuturor numelor de staţii cunoscute în întreaga lume — listă, de altfel, extrem de lungă.

Iată un alt exemplu de adăugare a unui utilizator pe un server MySQL folosind comanda GRANT, de această dată pentru a adăuga un utilizator numit jane cu parola 45sdg11, care să aibă toate (ALL) privilegiile pentru o tabelă numită employees din baza de date numită myCompany. Acest nou utilizator se va putea conecta doar de la o staţie predefinită:

```
GRANT ALL
ON myCompany.employees
TO jane@janescomputer.company.com
IDENTIFIED BY "45sdg11";
```

În cazul în care ştiţi că staţia janescomputer.company.com are adresa de IP 63.124.45.2, puteţi înlocui această adresă în porţiunea din comanda GRANT dedicată numelui staţiei (hostname), la fel ca în exemplul de mai jos:

```
GRANT ALL
ON myCompany.employees
TO jane@'63.124.45.2'
IDENTIFIED BY "45sdg11";
```

Trebuie avut în vedere un amănunt foarte important cu privire la adăugarea de utilizatori: folosiţi întotdeauna parole şi aveţi grijă ca aceste parole să îndeplinească cerinţele generale de securitate! Serverul MySQL vă permite să adăugaţi utilizatori fără a le defini o parolă, dar acest lucru lasă uşa deschisă acelor indivizi rău intenţionaţi care pot deduce un nume de utilizator din sistemul dumneavoastră care are toate privilegiile asociate.

În cazul în care folosiţi comanda `GRANT` pentru a adăuga utilizatori, modificările vor intra imediat în vigoare. Pentru a vă asigura complet de acest lucru, puteţi utiliza comanda `FLUSH PRIVILEGES` din interfaţa MySQL monitor, care duce la reîncărcarea tabelei de privilegii.

Revocarea privilegiilor

Revocarea privilegiilor este la fel de simplă ca şi adăugarea lor; singura deosebire este că în locul comenzii `GRANT` vom utiliza comanda `REVOKE`. Sintaxa comenzii `REVOKE` este următoarea:

```
REVOKE privileges
ON nume_baza_de_date.nume_tabela
FROM nume_utilizator@nume_host;
```

De asemenea, în acelaşi mod în care puteţi conferi privilegii utilizând comanda `INSERT` la nivelul tabelelor de sistem, le puteţi şi revoca folosind comanda `DELETE` pentru a şterge înregistrările aferente din tabelele de sistem din baza de date `mysql`. Totuşi, această metodă presupune o foarte bună cunoaştere a tabelelor şi câmpurilor aferente fiecărui privilegiu, de aceea recomandăm utilizarea comenzii `REVOKE`, aceasta oferind o metodă mult mai uşoară şi mai sigură de revocare a privilegiilor.

Pentru a revoca abilitatea utilizatorului `john` de a introduce înregistrări noi în baza de date `myCompany`, ar trebui să utilizaţi comanda `REVOKE`, aşa cum vă este ea prezentată în exemplul de mai jos:

```
REVOKE INSERT
ON myDB.*
FROM john@"%";
```

Modificările făcute cu ajutorul comenzii `REVOKE` intră în vigoare imediat, dar pentru ca serverul să afle despre aceste modificări trebuie să utilizaţi în interfaţa MySQL monitor comanda `FLUSH PRIVILEGES`.

Rezumat

Instalarea serverului MySQL pe platforma Windows este extrem de simplă datorită unei metode de instalare bazată pe utilitare de asistenţă. MySQL AB pune la dispoziţia utilizatorilor sistemului de operare Windows o interfaţă grafică de administrare, numită WinMySQLadmin. Utilizatorii de Linux/Unix nu beneficiază de o instalare bazată pe utilitare de asistenţă, dar nu este foarte dificil să urmeze o succesiune simplă de instrucţiuni care despachetează şi instalează fişierele binare ale serverului şi clientului MySQL. Utilizatorii unui sistem de operare de tipul Linux/Unix pot folosi şi pachete RPM pentru instalarea serverului MySQL.

Securitatea este întotdeauna o prioritate şi există o succesiune predefinită de paşi pe care puteţi să o urmaţi pentru a vă asigura că dispuneţi de o instalare sigură şi securizată. Chiar dacă nu sunteţi administratorul serverului, ar trebui să fiţi capabil să testaţi şi să identificaţi eventualele breşe de securitate, pe care să le sesizaţi ulterior administratorului serverului dumneavoastră MySQL.

Serviciul de server MySQL nu ar trebui să ruleze sub utilizatorul `root`. De asemenea, utilizatorii înregistraţi pe serverul MySQL ar trebui să aibă întotdeauna o parolă, iar privilegiile lor de acces ar trebui să fie întotdeauna foarte bine definite.

MySQL foloseşte tabelele de privilegii într-un proces în doi paşi pentru fiecare solicitare făcută. MySQL trebuie să ştie cine sunteţi şi de unde vă conectaţi şi fiecare porţiune de informaţie pe care o oferiţi în cadrul acestui proces trebuie să se corespundă cu informaţiile stocate în baza de date `mysql`, în tabelele de privilegii. De asemenea, utilizatorul trebuie să beneficieze de permisiuni explicite pentru a executa un anumit tip de comandă.

Puteţi adăuga utilizatori prin intermediul comenzii `GRANT`, care utilizează o sintaxă simplă pentru a adăuga intrări în tabela `user` din baza de date `mysql`. Comanda `REVOKE`, cu o sintaxă la fel de simplă, se foloseşte pentru a şterge aceste privilegii.

Întrebări şi răspunsuri

Î. ***Cum şterg complet un utilizator de pe server? Comanda*** `REVOKE` ***şterge doar privilegiile.***

R. Pentru a elimina complet un utilizator de pe server trebuie să utilizaţi comanda `DELETE` asupra înregistrării corespunzătoare acelui utilizator din tabela `user` a bazei de date `mysql`.

Î. ***Ce pot să fac dacă îl rog pe furnizorul meu de servicii Internet să nu mai ruleze serverul de MySQL sub utilizatorul*** `root`***, iar acesta nu dă curs solicitării mele?***

R. Schimbaţi-vă furnizorul. Dacă furnizorul dumneavoastră de servicii Internet nu recunoaşte riscul de a rula ceva atât de important precum serverul dumneavoastră de baze de date sub utilizatorul `root` şi nu dă curs cererii dumneavoastră, găsiţi-vă alt furnizor. În acest moment există pe piaţă furnizori care, pentru sume derizorii de genul 9,95 USD pe lună, nu rulează procese importante sub utilizatorul `root`.

Atelier

Atelierul a fost special conceput pentru a anticipa eventualele întrebări din partea dumneavoastră, pentru a vă permite să recapitulaţi ceea ce aţi învăţat pe parcursul acestei lecţii şi pentru a vă permite să începeţi să puneţi în practică cunoştinţele proaspăt achiziţionate.

Test

1. Adevărat sau fals: aplicaţia Telnet este o metodă foarte bună de a vă conecta la serverul MySQL de la un calculator aflat la distanţă, respectând cerinţele de securitate.

2. Care sunt cele trei informaţii pe care serverul MySQL le verifică la fiecare interogare?

3. Ce comandă aţi folosi pentru a conferi privilegiile `SELECT`, `INSERT` şi `UPDATE` unui utilizator numit `bill` de la staţia `localhost`, pentru toate tabelele din baza de date `BillDB`? De asemenea, ce informaţie lipseşte din această comandă şi este recomandată din motive de securitate?

Soluţii

1. Fals. Cuvântul cheie este securitate, iar aplicaţia Telnet nu foloseşte criptarea transmisiunilor de date între două staţii. Folosiţi SSH în locul său.

2. Cine sunteţi, de unde accesaţi serverul şi ce acţiuni aveţi dreptul să iniţiaţi.
3. Comanda este:

```
GRANT SELECT, INSERT, UPDATE
ON BillDB.*
TO bill@localhost;
```

Informaţia foarte importantă care lipseşte este parola.

Activităţi

1. Gândiţi-vă la situaţii în care aţi dori să restricţionaţi privilegiile de execuţie a comenzilor la nivel de tabelă. De exemplu, nu v-aţi dori ca personalul de birou să aibă privilegii de oprire a serverului de baze de date al întregii corporaţii.
2. Dacă aveţi drepturi administrative pe serverul MySQL, folosiţi comanda `GRANT` pentru a crea câţiva utilizatori de test. Nu contează dacă tabelele sau bazele de date pe care le utilizaţi există sau nu.
3. Utilizaţi comanda `REVOKE` pentru a revoca o parte dintre privilegiile utilizatorilor creaţi la pasul 2.

CAPITOLUL 2
Instalarea şi configurarea serverului Apache

În acest al doilea capitol dintre cele trei legate de instalare şi configurare, veţi avea ocazia să instalaţi serverul Web Apache şi să vă familiarizaţi cu principalele sale componente, inclusiv jurnalele şi fişierele de configurare.

În acest capitol, veţi învăţa:

- cum să instalaţi serverul Apache pe un sistem Linux/Unix;
- cum să instalaţi serverul Apache pe un sistem Windows;
- cum să modificaţi configurările serverului Apache;
- unde pot fi găsite fişierele de configurare şi jurnalele serverului Apache.

Versiuni curente şi viitoare ale serverului Apache

Instrucţiunile de instalare din acest capitol sunt specifice versiunii 2.0.47, care este versiunea curentă de producţie a acestui software. Fundaţia Apache (Apache Software Foundation) foloseşte numărul de pachet pentru versiuni conţinând îmbunătăţiri ale securităţii sau rezolvări de bug-uri. Pachetele nu sunt puse la dispoziţia publicului după un program bine determinat şi prestabilit; când sunt adăugate îmbunătăţiri sau rezolvări de probleme şi testate corespunzător, Fundaţia Apache publică o nouă versiune, cu un nou număr de pachet.

Există posibilitatea ca în momentul în care citiţi această carte numărul de versiune să se fi schimbat în 2.0.48 sau chiar mai mult. În acest caz ar trebui să consultaţi lista cu schimbări disponibilă la adresa Web `http://httpd.apache.org/download.cgi` şi să urmăriţi eventualele diferenţe în ceea ce priveşte procesul de instalare şi configurare, care este subiectul acestui capitol.

Deşi este puţin probabil ca instrucţiunile să se schimbe substanţial de la o versiune la alta, ar trebui să vă formaţi obiceiul de a consulta mereu lista de schimbări pentru aplicaţiile pe care le instalaţi şi administraţi. Dacă versiunea evoluează până la momentul în care citiţi această carte, dar nu sunt înregistrate

schimbări legate de procesul de instalare, amintiţi-vă pur şi simplu să substituiţi noul număr de versiune oriunde apare acesta în instrucţiunile de instalare sau în figurile care le însoţesc.

Alegerea unei metode de instalare

În ceea ce priveşte o instalare de bază pentru serverul Apache, aveţi mai multe opţiuni. Serverul Apache este distribuit în sistem *open source*, ceea ce înseamnă că aveţi acces la codul sursă al software-ului, acest lucru permiţându-vă literalmente să vă compilaţi propriul dumneavoastră server Apache personalizat. De asemenea, la dispoziţia publicului sunt puse şi distribuţii binare ale serverului Apache, pentru toate platformele moderne de tip Linux/Unix. Exemplele din acest capitol vă vor învăţa cum să compilaţi serverul Apache folosind codurile sursă dacă folosiţi o platformă de tip Linux/Unix şi cum să folosiţi programul de instalare, dacă plănuiţi să folosiţi serverul Apache într-un mediu de dezvoltare Windows.

Apropo

Pentru detalii despre instalarea serverului Apache folosind CD-ROM-ul care însoţeşte această carte, vă rugăm să consultaţi anexa A, *Instalarea aplicaţiilor MySQL, Apache şi PHP aflate pe CD-ROM-ul ataşat cărţii.*

Compilarea surselor

Compilarea surselor vă oferă cea mai mare flexibilitate pentru că vă permite să vă construiţi propriul dumneavoastră server personalizat, prin eliminarea modulelor inutile şi să vă îmbunătăţiţi serverul prin adăugarea de module noi. Compilarea serverului Apache folosind codul sursă vă permite să actualizaţi serverul în permanenţă la ultima versiune disponibilă, precum şi să aplicaţi patch-uri de securitate, în timp ce versiunile noi oferite sub formă de distribuţii binare pot să apară abia după câteva zile sau chiar săptămâni.

Procesul de compilare a codului sursă pentru serverul Apache nu este deloc dificil, mai ales pentru instalările minimale, dar poate deveni din ce în ce mai complex şi mai laborios pe măsură ce sunt implicate module sau biblioteci oferite de terţi.

Instalarea folosind fişiere binare

Fişierele de instalare binare pentru platformele Linux/Unix sunt disponibile de la diferiţi vânzători şi pot fi, de asemenea, descărcate de pe site-ul Web al Fundaţiei Apache. Aceste fişiere sunt deosebit de utile, mai ales pentru utilizatorii fără cunoştinţe de administrare sau fără nevoi speciale de configurare. Există furnizori comerciali, independenţi de Fundaţia Apache, care oferă pachete preconfigurate ce conţin aplicaţia server, modulele de bază, suport pentru adăugarea de noi module, asistenţă tehnică şi alte facilităţi.

Fundaţia Apache oferă un program de instalare pentru sistemele bazate pe platforma Windows — o platformă pentru care un compilator nu este întotdeauna disponibil, aşa cum se întâmplă pe platformele de tip Linux/Unix.

Instalarea serverului Web Apache pe platforma Linux/Unix

Această secţiune explică modalitatea de instalare a unei versiuni noi de server Apache 2.0.x pe platforma Linux/Unix. Paşii necesari pentru o instalare reuşită folosind compilarea codurilor sursă a serverului Apache sunt:

1. descărcarea aplicaţiei software;
2. rularea fişierului de configurare;
3. compilarea codului sursă şi instalarea lui.

Următoarea secţiune descrie detaliat aceşti paşi.

Descărcarea codurilor sursă a serverului Apache

Site-ul oficial Apache, de unde puteţi descărca aceste pachete, este accesibil la adresa `http://httpd.apache.org/`. Pe acest site puteţi găsi diferite versiuni ale serverului Apache, comprimate folosind diferite metode de compresie. Fişierele din distribuţie sunt împachetate iniţial cu ajutorul utilitarului `tar`, ulterior fiind comprimate folosind fie utilitarul `gzip`, fie utilitarul `compress`. Descărcaţi versiunea `.tar.gz` dacă aveţi utilitarul `gunzip` instalat pe sistemul dumneavoastră. Acest utilitar este, de obicei, instalat o dată cu sistemul de operare pentru platformele open source precum FreeBSD sau Linux. Descărcaţi fişierele `tar.Z` dacă nu aveţi utilitarul `gunzip` instalat pe sistemul dumneavoastră. (În cazul multor versiuni comerciale ale platformei Unix, acesta nu este instalat iniţial.)

Fişierul de care aveţi nevoie ar trebui să fie denumit în genul `httpd-2.0.versiune.tar.z` sau `httpd-2.0.versiune.tar.gz`, unde `versiune` este cea mai nouă versiune a serverului Apache. De exemplu, versiunea 2.0.47 a serverului Apache este disponibilă pentru descărcare sub forma fişierului numit `httpd-2.0.47.tar.gz`. Păstraţi fişierele descărcate într-un director rezervat pentru fişiere care conţin coduri sursă precum `/usr/src/` sau `/usr/local/src/`.

Decomprimarea codurilor sursă a serverului Web Apache

Dacă aţi descărcat pachetul tarball comprimat cu utilitarul `gzip` (fişierul are terminaţia `.tar.gz`), îl puteţi decomprima folosind utilitarul `gunzip` (care face parte din distribuţia standard `gzip`).

Tarball este o poreclă uzuală a pachetelor create cu ajutorul utilitarului `tar`.

Apropo

Puteţi decomprima şi despacheta fişierul tastând următoarea comandă:

```
# gunzip < httpd-2.0*.tar.gz | tar xvf -
```

În cazul în care aţi descărcat fişierul tar comprimat folosind utilitarul compress (caz în care fişierul are terminaţia .tar.z), îl puteţi decomprima şi despacheta tastând următoarea comandă:

```
# cat httpd-2.0*.tar.Z | uncompress | tar xvf -
```

Decomprimarea pachetului tarball creează o structură de directoare, directorul de nivel superior fiind denumit httpd-2.0_versiune. Accesaţi acest director pentru a vă pregăti pentru configurarea şi instalarea aplicaţiei.

Pregătiri pentru instalare

Puteţi specifica ce caracteristici va avea fişierul binar rezultat în urma procesului de compilare folosind scriptul configure din directorul de nivel superior. Standard, Apache va fi compilat cu un set normal de module compilate static şi va fi instalat în directorul /usr/local/apache2. Dacă sunteţi mulţumit de aceste configurări puteţi tasta următoarea comandă pentru a configura serverul Apache:

```
#./configure
```

Totuşi, pentru a pregăti instalarea PHP, care va fi prezentată în capitolul 3, va trebui să vă asiguraţi că fişierul este compilat cu modulul mod_so activat. Acest modul, denumit după formatul Unix shared object (*.so — obiecte Unix partajate), permite utilizarea în cadrul serverului Web a modulelor dinamice precum PHP. Pentru a configura serverul Apache în aşa fel încât el să se instaleze într-o anumită locaţie (în cazul nostru /usr/local/apache2/) şi pentru a activa opţiunea de încărcare a modulului dinamic mod_so, tastaţi următoarea comandă:

```
#./configure --prefix=/usr/local/apache2 --enable-module=so
```

Scopul acestui script configure este să determine toate detaliile legate de prezenţa în sistem a bibliotecilor necesare, opţiuni de compilare, diferenţe specifice platformei de operare şi altele, precum şi de a crea un set de fişiere speciale numite makefiles. Fişierele makefile conţin instrucţiuni care permit efectuarea diferitor operaţiuni numite targets, precum compilarea fişierului binar (executabil) al serverului Apache. Aceste fişiere vor fi citite de utilitarul make, care va efectua aceste operaţiuni. Dacă totul decurge conform aşteptărilor, după rularea scriptului configure, veţi putea vedea un set de mesaje legate de diferite verificări pe care acesta le efectuează şi veţi fi readus la prompterul liniei de comandă:

```
...
creating test/Makefile
config.status: creating docs/conf/httpd-std.conf
config.status: creating include/ap_config_layout.h
config.status: creating support/apxs
config.status: creating support/apachectl
config.status: creating support/dbmmanage
config.status: creating support/envvars-std
config.status: creating support/log_server_status
config.status: creating support/logresolve.pl
```

```
config.status: creating support/phf_abuse_log.cgi
config.status: creating support/split-logfile
config.status: creating build/rules.mk
config.status: creating include/ap_config_auto.h
config.status: executing default commands
#
```

Dacă scriptul de configurare eşuează, pe ecran vor apărea diferite mesaje de atenţionare, care vă avertizează să instalaţi diferite pachete software necesare instalării serverului Apache, precum compilatoare sau biblioteci. După ce instalaţi aplicaţiile software lipsă puteţi rula din nou comanda `configure`, după ce în prealabil veţi fi şters fişierele `config.log` şi `config.status` din directorul de nivel superior.

Compilarea şi instalarea serverului Apache

Utilitarul `make` citeşte informaţiile stocate în fişierele `makefile` şi compilează executabilul server şi modulele. Tastaţi `make` în linia de comandă pentru a compila serverul Apache. Veţi vedea o serie de mesaje care indică progresele compilării, iar la final veţi vedea, din nou, prompterul liniei de comandă. După ce procesul de compilare s-a terminat, puteţi trece mai departe la instalarea efectivă a serverului Apache, tastând `make install` în linia de comandă. Fişierele `makefile` vor instala toate fişierele şi vor crea directoarele, la finalul procesului reîntorcându-se la prompterul liniei de comandă:

```
...
Installing header files
Installing man pages and online manual
mkdir /usr/local/apache2/man
mkdir /usr/local/apache2/man/man1
mkdir /usr/local/apache2/man/man8
mkdir /usr/local/apache2/manual
Installing build system files
make[1]: Leaving directory `/usr/local/src/httpd-2.0.40'
#
```

Fişierele distribuţiei Apache ar trebui să fie acum în directorul `/usr/local/apache2`, aşa cum a fost specificat de către parametrul `--prefix` din comanda `configure`. Pentru a verifica dacă fişierul executabil `httpd` a fost corect compilat tastaţi următoarea comandă:

```
# /usr/local/apache2/bin/httpd -v
```

Ar trebui să vedeţi următoarele linii (versiunea ar putea fi diferită):

```
Server version: Apache/2.0.47
Server built:   Sep  28 2003 11:47:22
```

Dacă doriţi să învăţaţi cum se instalează serverul Apache pe platforma Windows, continuaţi. Dacă nu, puteţi trece direct la secţiunea intitulată *Structura fişierului de configurare a serverului Apache*, aflată la pagina 35, pentru a învăţa despre acesta.

Instalarea serverului Web Apache sub sistemul de operare Windows

Serverul Apache 2.0 rulează pe majoritatea sistemelor de operare Windows şi oferă performanţe şi stabilitate vizibil îmbunătăţite pentru această platformă începând cu versiunile 1.3. Puteţi să compilaţi executabilele cu ajutorul codurilor sursă, dar pentru că nu foarte multe sisteme de operare Windows sunt echipate cu un compilator, această secţiune tratează instalarea cu ajutorul programului de instalare.

Înainte de a instala serverul Apache, veţi dori, probabil, să verificaţi dacă nu rulaţi deja un server Web (de exemplu, o versiune anterioară de Apache sau Microsoft Internet Information Server sau Microsoft Personal Web Server). În caz că descoperiţi acest lucru, ar fi o idee bună să dezinstalaţi sau să dezactivaţi serverele existente. Există posibilitatea de a rula mai multe servere web în acelaşi timp, dar ele ar trebui să ruleze pe combinaţii diferite de adrese IP şi porturi.

Înainte de a trece la descărcarea programului de instalare, vă rugăm să vă opriţi un moment asupra atenţionărilor de pe pagina de descărcare (pe care o găsiţi la adresa `http://httpd.apache.org/download.cgi`) şi care spune `If you are downloading the Win32 distribution, please read these important notes` (dacă descărcaţi distribuţia Win32, vă rugăm să citiţi aceste note foarte importante). Adresa directă pentru această pagină este `http://www.apache.org/dist/httpd/binaries/win32/README.html`.

Fundaţia Apache întreţine această pagină spre beneficiul tuturor utilizatorilor de sisteme de operare Windows care doresc să ruleze o versiune de server Web Apache. Pe această pagină există note pentru aproape orice versiune de sistem de operare Windows încă folosit şi este în interesul dumneavoastră să citiţi informaţiile prezentate aici. Garantăm că, dacă rulaţi Apache, fie ca server de producţie, fie ca server de dezvoltare, veţi găsi informaţii foarte importante pe această pagină. O dată ce sunteţi gata să începeţi instalarea, căutaţi legătura numită Win32 Binary (MSI Installer). După ce aţi descărcat programul de instalare, daţi clic dublu pe fişier pentru a începe procesul de instalare. Veţi fi întâmpinaţi de un ecran de bun venit, aşa cum apare în figura 2.1. Selectaţi Next pentru a continua procesul de instalare şi veţi fi rugaţi să acceptaţi licenţa Apache. În linii mari, licenţa stipulează că puteţi face orice doriţi cu aplicaţia software, inclusiv să executaţi modificări personale, mai puţin să pretindeţi că dumneavoastră aţi scris acest program. Totuşi, ar fi o idee bună să citiţi în întregime licenţa pentru a fi siguri că aţi înţeles termenii ei.

După ce acceptaţi licenţa, programul de instalare vă prezintă o scurtă introducere a serverului Web Apache. Imediat după aceea, vă roagă să introduceţi o descriere sumară a calculatorului dumneavostră, aşa cum se vede în figura 2.2. Această descriere conţine adresa completă de reţea a serverului (de exemplu, `calc_meu.domeniul_meu.com`) şi adresa de e-mail a administratorului serverului. Numele serverului va fi numele pe care îl vor folosi clienţii dumneavoastră pentru a accesa resursele acestuia, iar adresa de e-mail a administratorului va fi adăugată mesajelor de eroare astfel încât vizitatorii să ştie cum să vă contacteze atunci când ceva nu funcţionează corespunzător.

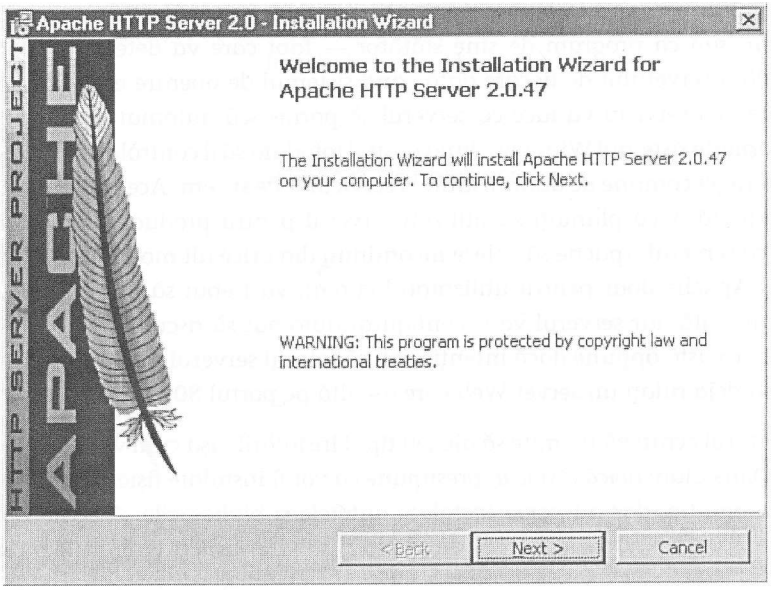


FIGURA 2.1. Ecranul de întâmpinare al programului de instalare.

Apropo

Dacă maşina dumneavoastră nu are o adresă de reţea completă, folosiţi `localhost` sau `127.0.0.1` drept nume al serverului.

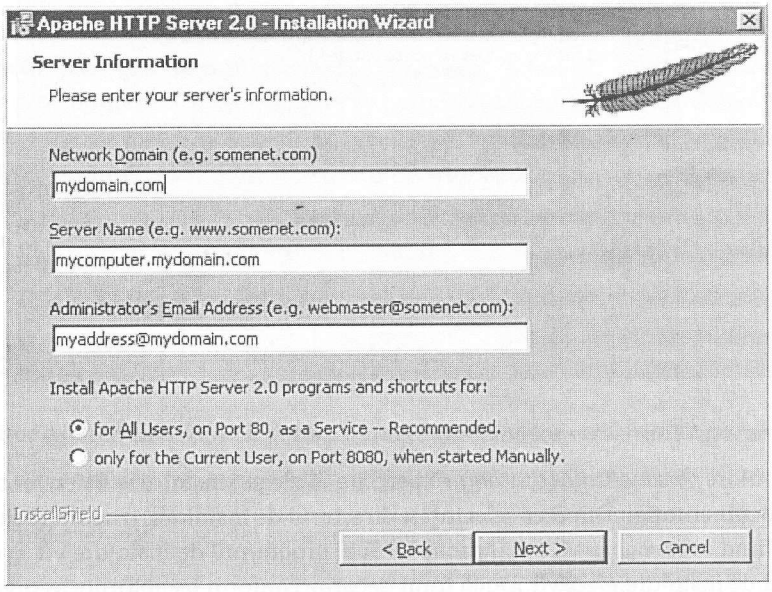


FIGURA 2.2. Ecranul cu informaţii despre server.

Următorul pas vă invită să alegeţi între a instala serverul Apache ca serviciu de sistem sau ca program de sine stătător — fapt care va determina pornirea manuală a serverului de fiecare dată când sistemul de operare este reiniţializat. Instalarea ca serviciu va face ca serverul să pornească automat de fiecare dată când porneşte sistemul Windows, fiind posibil totodată să îl controlaţi prin intermediul interfeţei comune de administrare a serviciilor de sistem. Această opţiune este recomandată dacă plănuiţi să utilizaţi serverul pentru producţie sau dacă este nevoie ca serverul Apache să ruleze încontinuu din orice alt motiv. Dacă instalaţi serverul Apache doar pentru utilizatorul curent, va trebui să îl porniţi manual de fiecare dată, iar serverul va fi configurat automat să asculte pe portul 8080. Selectaţi această opţiune dacă intenţionaţi să folosiţi serverul Apache pentru teste sau dacă deja rulaţi un server Web care ascultă pe portul 80.

Următorul ecran vă permite să alegeţi tipul instalării, aşa cum vă apare în figura 2.3. O instalare *tipică* (*Typical*) presupune că vor fi instalate fişierele executabile şi documentaţia, dar nu vor fi instalate antetele şi bibliotecile. Această opţiune este recomandată în cazul în care nu plănuiţi să vă compilaţi propriile module.

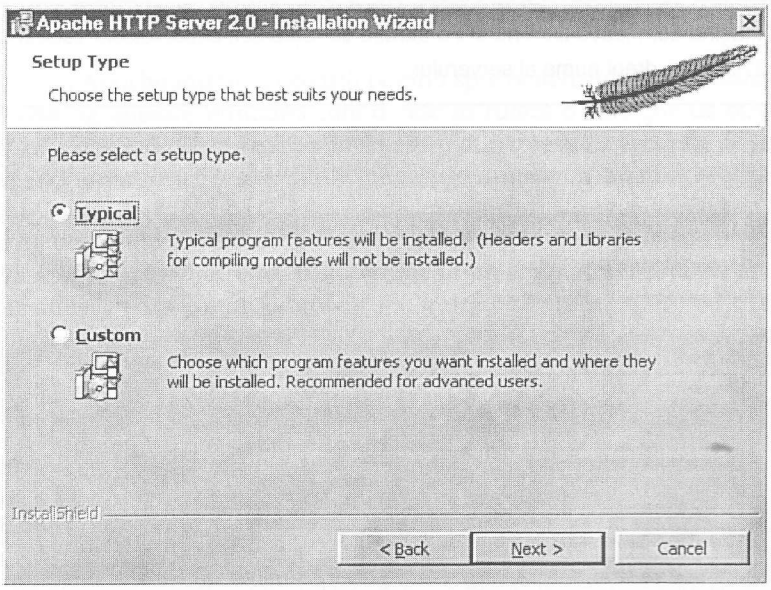


FIGURA 2.3. Ecranul de selectare a metodei de instalare.

O instalare *personalizată* (*Custom*) vă permite să alegeţi instalarea fişierelor antet sau a documentaţiei. După ce selectaţi şi directorul de instalare, care implicit este ales ca fiind `c:\Program Files\Apache Group`, programul de instalare va trece la procesul de instalare efectivă. Dacă totul decurge conform aşteptărilor veţi vedea ecranul final, aşa cum apare în figura 2.4.

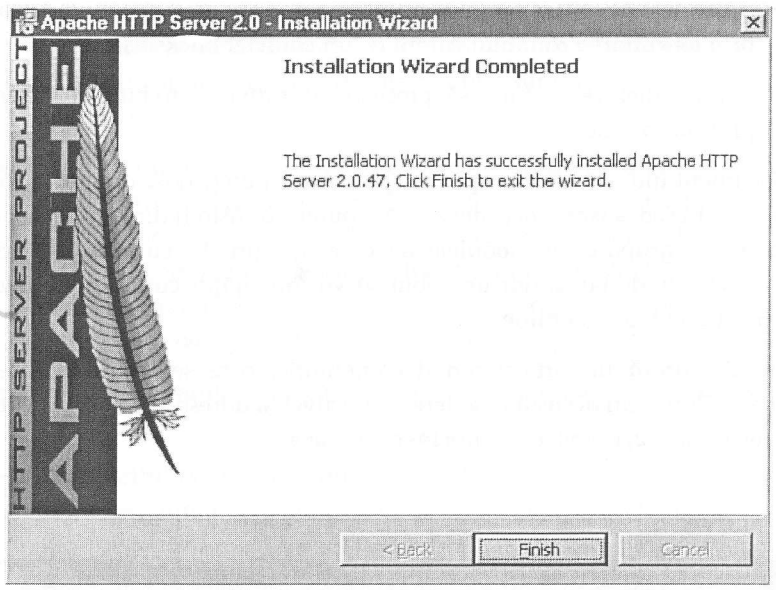


FIGURA 2.4. Ecranul de instalare încununată de succes.

În următoarea secţiune veţi învăţa mai multe despre fişierul de configurare al serverului Apache şi, în cele din urmă, veţi porni serverul pe care tocmai l-aţi instalat.

Structura fişierului de configurare a serverului Apache

Serverul Apache îşi păstrează toate informaţiile de configurare în fişiere de tip text. Fişierul principal de configurare se numeşte `httpd.conf`. Acest fişier conţine directive şi containere de directive, care vă permit să personalizaţi instalarea dumneavoastră a serverului Apache. Directivele conţin informaţii despre configurările specifice serverului Apache, precum autorizări, performanţă sau parametri de reţea. Containerele specifică contextul în care se aplică acele configurări. De exemplu, configurările privind autorizarea pot să se refere la întregul server, la un director sau la un singur fişier.

Directivele

Următoarele reguli se aplică pentru sintaxa directivelor serverului Apache:

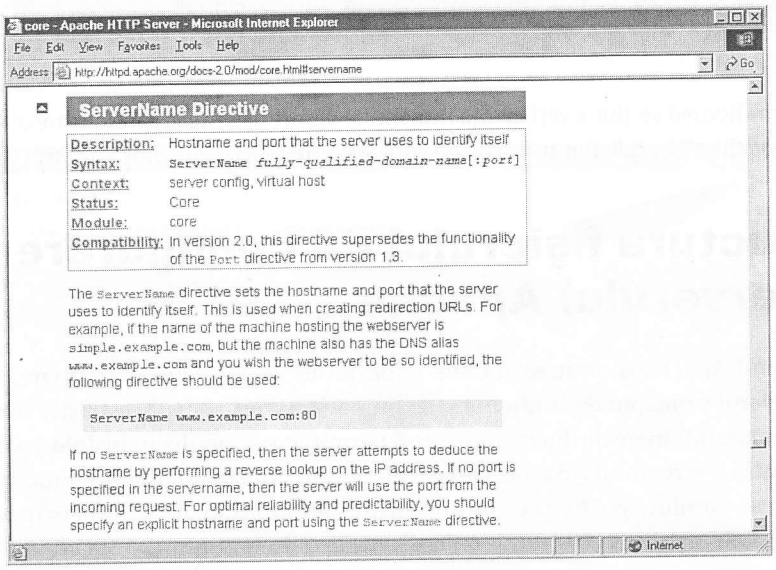
- argumentele directivei urmează după numele directivei;
- argumentele directivei sunt separate de spaţii;
- numărul şi tipul argumentelor variază de la o directivă la alta; unele dintre ele nu au nici un argument;

- o directivă ocupă un singur rând, dar poate continua şi pe rândul următor prin terminarea rândului curent cu un caracter backslash (\);
- semnul diez (#) trebuie să preceadă directiva şi trebuie să apară pe propria sa linie.

În documentaţia serverului Apache, pe care o puteţi găsi online la adresa Web `http://httpd.apache.org/docs-2.0/`, puteţi să răsfoiţi directivele în ordine alfabetică sau grupate pe modulele de care aparţin. În curând, veţi învăţa despre directivele de bază, dar ar trebui să vă îmbogăţiţi cunoştinţele folosind documentaţia publicată online.

Figura 2.5 arată un articol din documentaţie, care se referă la directiva `ServerName`. Puteţi citi această descriere a directivei la adresa Web `http://httpd.apache.org/docs-2.0/mod/core.html#servername`.

FIGURA 2.5. Exemplu de descriere a directivei.



Schema, aşa cum vă este detaliată în documentaţie la adresa Web `http://httpd.apache.org/docs-2.0/mod/directive-dict.html`, este aceeaşi pentru toate directivele.

- **Syntax (sintaxă)** — Această intrare explică formatul opţiunilor directivei. Parametrii obligatorii apar cu litere cursive, parametri opţionali apărând cu caractere cursive între paranteze mari.
- **Default (valoare implicită)** — Dacă directiva are o valoare implicită, aceasta va apărea aici.

- **Context (context)** — Această intrare detaliază secţiunile sau containerele în care poate apărea directiva. Containerele vă sunt explicate pe larg în secţiunea următoare. Valorile posibile sunt `server config`, `virtual host`, `directory`, şi `.htaccess`.
- **Status (stare)** — Această intrare indică dacă directiva este predefinită în nucleul Apache, dacă aparţine unuia din modulele furnizate în pachetul Apache (`base` pentru modulele compilate implicit sau `extension` pentru modulele compilate prin intervenţia utilizatorului), dacă este parte a unui modul de multiprocesare (`MPM`) sau dacă, deşi se găseşte în pachetul Apache, nu este încă gata de utilizare într-un mediu de producţie (`experimental`).
- **Module (modul)** — Această intrare indică modulul căruia i se aplică directiva.
- **Compatibility (compatibilitate)** — Această intrare conţine informaţii despre versiunile serverului Apache care suportă directiva.
- **Override (contrazicere)** — Directivele serverului Apache aparţin diferitelor categorii. Câmpul Override este folosit pentru a specifica ce categorii de directive pot să apară în `.htaccess` — fişierele de configurare per director.

În documentaţie, articolele sunt urmate de o scurtă explicaţie a directivelor, precum şi de referinţe către directive asemănătoare sau documentaţie suplimentară. Acestea pot fi găsite la finalul articolului.

Containerele

Containerele de directive, cunoscute şi sub numele de secţiuni (`sections`), limitează domeniul de aplicabilitate a directivelor. Dacă directivele nu sunt găzduite într-un container, atunci ele aparţin domeniului general al serverului (`server config`) şi se aplică întregului server.

Iată containerele standard ale fişierului de configurare al serverului Apache:

- `<VirtualHost>` — O directivă `VirtualHost` specifică un host virtual (gazdă virtuală). Serverul Apache vă permite să găzduiţi mai multe site-uri Web pe un singur server. Directivele din acest container se aplică unuia dintre serverele găzduite în acest fel. Această directivă acceptă drept argumente un nume de domeniu sau o adresă IP şi, opţional, un număr de port. Veţi învăţa mai multe despre host-urile virtuale în capitolul 26, intitulat *Reglarea performanţei serverului Apache şi folosirea host-urilor virtuale.*
- `<Directory>`, `<DirectoryMatch>` — Aceste containere permit ca directivele să se aplice doar asupra unui director sau grup de directoare din sistemul de fişiere. Containerul `Directory` acceptă ca argument un nume sau o expresie echivalentă unor nume de directoare. Directivele conţinute se referă la directoarele specificate şi la subdirectoarele lor. Containerul `DirectoryMatch` acceptă ca argument şi expresii regulate. De exemplu,

următoarea linie permite găsirea tuturor directoarelor de nivel doi ale directorului rădăcină www, care au numele format din patru cifre, cum ar fi, de exemplu, un director numit după o lună şi un an (Ø9Ø2 pentru septembrie 2002):

```
<DirectoryMatch "^/www/.*/[Ø-9]{4}">
```

- `<Location>`, `<LocationMatch>` — Aceste containere permit ca directivele incluse să se aplice anumitor adrese URL sau unor adrese URL specificate prin intermediul unor expresii. Ele sunt similare omoloagelor lor care tratează directoarele, expresiile de mai jos aplicându-se atât pentru calea `"/my/data"`, cât şi pentru calea `"/your/data"`:

```
<LocationMatch "/(my|your)/data">
```

- `<Files>`, `<FilesMatch>` — La fel ca şi containerele `Directory` şi `Location`, secţiunile `Files` le permit directivelor să se aplice doar anumitor fişiere sau unor fişiere specificate prin intermediul unor expresii.

Directivele se scriu între marcajele de container, aşa cum se vede în listingul 2.1.

Listingul 2.1. Exemplu de directive în interiorul containerelor.

```
 1: <Directory "/o/directiva">
 2: ODirectivaOarecare1
 3: ODirectivaOarecare2
 4: </Directory>
 5: <Location "/downloads/*.html">
 6: ODirectivaOarecare3
 7: </Location>
 8: <Files "\.(gif|jpg)">
 9: ODirectivaOarecare4
1Ø: </Files>
```

Directivele *ODirectivaOarecare1* şi *ODirectivaOarecare2*, date de exemplu, se vor aplica directorului `/www/docs` şi tuturor subdirectoarelor acestuia. Directiva *ODirectivaOarecare3* se va aplica acelor adrese URL care se referă la pagini cu extensia `.html`, din adresa URL `/download/`. Directiva *ODirectivaOarecare4* se va aplica tuturor fişierelor cu extensia `.gif` şi `.jpg`.

Evaluarea condiţiilor

Serverul Apache oferă posibilitatea definirii de containere condiţionale. Directivele incluse în acest tip de containere vor fi procesate doar dacă anumite condiţii sunt îndeplinite.

- <IfDefine> — Directivele din acest container vor fi procesate doar dacă executabilul Apache primeşte un anumit parametru prin linia de comandă. Directiva din listingul 2.2 va fi procesată doar dacă parametrul `-DMyModule` este transmis executabilului Apache. Puteţi să transmiteţi direct acest parametru sau să modificaţi scriptul `apachectl`, aşa cum vă este explicat în secţiunea intitulată *Comenzi ce pot fi trimise serverului Apache* pe care o veţi întâlni începând de la pagina 42 în acest capitol.

Containerul `IfDefine` permite şi folosirea de expresii negative. Aceasta înseamnă că directivele din interiorul unui container `<IfDefine !MyModule>` vor fi procesate doar dacă parametrul `-DMyModule` nu a fost transmis prin linia de comandă executabilului Apache. De exemplu, dacă `-DSSL` nu este transmis prin linia de comandă ca parametru al executabilului Apache, serverul nu va asculta pe portul SSL (de obicei 443).

- `<IfModule>` — Directivele dintr-un container `IfModule` vor fi procesate doar dacă modulul transmis ca argument executabilului Apache este prezent pe serverul Web. De exemplu, serverul Apache este pus la dispoziţia publicului cu un fişier de configurare `httpd.conf` care conţine configurări pentru diferite module MPM. Doar configurările care aparţin modulelor MPM ce au fost compilate o dată cu executabilul vor fi procesate, aşa cum puteţi vedea în listingul 2.3. Scopul acestui exemplu este să ilustreze faptul că va fi evaluat doar un grup de directive.

Listingul 2.2. Exemplu de container `IfDefine`.

```
1: <IfDefine MyModule>
2: LoadModule my_module modules/libmymodule.so
3: </IfDefine>
```

Listingul 2.3. Exemplu de container `IfModule`.

```
 1: <IfModule prefork.c>
 2: StartServers         5
 3: MinSpareServers      5
 4: MaxSpareServers     10
 5: MaxClients          20
 6: MaxRequestsPerChild  0
 7: </IfModule>
 8:
 9: <IfModule worker.c>
10: StartServers         3
11: MaxClients           8
12: MinSpareThreads      5
13: MaxSpareThreads     10
14: ThreadsPerChild     25
15: MaxRequestsPerChild  0
16: </IfModule>
```

Directiva `ServerRoot`

Directiva `ServerRoot` acceptă un singur argument: o cale de director care indică directorul în care se găseşte serverul. Toate căile relative folosite în alte directive sunt evaluate faţă de calea absolută definită de directiva ServerRoot. Dacă aţi compilat serverul Apache folosind codurile sursă pe platforma Linux/Unix, aşa cum v-a fost prezentat la începutul acestui capitol, valoarea implicită a directivei ServerRoot este `/usr/local/apache2`. Dacă aţi folosit programul de instalare pe platforma Windows, valoarea implicită pentru ServerRoot va fi `c:\Program Files\Apache Group`.

Fişierele de configurare per director

Serverul Apache foloseşte fişierele de configurare per director pentru a permite introducerea directivelor în afara fişierului de configurare principal, `httpd.conf`. Aceste fişiere speciale pot fi plasate în interiorul sistemului de fişiere. Serverul Apache va procesa aceste fişiere atunci când este solicitat un document care se găseşte într-unul dintre aceste directoare sau în subdirectoarele sale. Toate fişierele de configurare per director întâlnite în cale sunt concatenate şi procesate. De exemplu, dacă serverul Apache primeşte o solicitare pentru fişierul `/usr/local/apache2/htdocs/index.html`, acesta va căuta toate fişierele de configurare per director din directoarele `/`, `/usr`, `/usr/local`, `/usr/local/apache2`, în ordinea de mai sus.

Folosirea fişierelor de configurare per director duce la o scădere a performanţei. Serverul Apache trebuie să efectueze operaţiuni de căutare pe hard disc pentru fiecare solicitare, chiar dacă respectivele fişiere de configurare per director nu există.

Fişierele de configurare per director se numesc implicit `.htaccess`. Acest nume are o proveniență istorică — fişierele astfel denumite erau folosite pentru restricţionarea accesului la directoarele care conţineau fişierele HTML.

Directiva `AccessFileName` vă permite să schimbaţi numele fişierelor de configurare per director din `.htaccess` în orice doriţi. Această directivă acceptă o listă a tuturor fişierelor pe care serverul Apache le va căuta şi procesa atunci când trebuie să găsească fişierele de configurare per-director. Pentru a stabili dacă o directivă globală poate fi contrazisă de o directivă dintr-un fişier de configurare per director verificaţi dacă în câmpul `Context:` al sintaxei directivei respective există valoarea `.htaccess`.

Directivele serverului Apache aparţin unor grupuri diferite, specificate prin intermediul câmpului Override:, din descrierea sintaxei directivei. Valorile posibile sunt:

- `AuthConfig` — directive de autorizare.
- `FileInfo` — directive care controlează tipul documentelor.
- `Indexes` — directive care controlează indexarea directoarelor.
- `Limit` — directive care controlează accesul pe baza adresei.
- `Options` — directive care controlează diferite caracteristici ale directoarelor.

Puteţi controla care dintre aceste grupuri de directive pot apărea în fişierele de configurare per director, prin folosirea directivei `AllowOverride`. Directiva `AllowOverride` poate accepta, de asemenea, şi argumentul `All` (toate) sau `None` (nici una). `All` înseamnă că toate directivele din toate grupurile pot să apară

în fişierele de configurare per director. None dezactivează utilizarea fişierelor de configurare per director pentru un anumit director şi pentru toate subdirectoarele sale. Listingul 2.4 vă arată cum să dezactivaţi fişierele de configurare per director pentru întregul server. Acest lucru îmbunătăţeşte simţitor performanţele serverului şi reprezintă configurarea implicită a serverului Apache.

Listingul 2.4. Dezactivarea fişierelor de configurare per director.

```
1: <Directory />
2: AllowOverride none
3: </Directory>
```

Fişierele jurnal ale serverului Apache

Serverul Apache include, implicit, două fişiere jurnal. Fişierul access_log este folosit pentru a urmări solicitările clienţilor. Fişierul error_log este folosit pentru a înregistra evenimente importante, cum ar fi erorile apărute pe server sau repornirile serverului. Aceste fişiere nu există până în momentul primei porniri a serverului. Aceste fişiere sunt numite access.log şi error.log în cazul platformelor Windows.

Fişierul access_log

Când un client solicită un anumit fişier de pe server, Apache înregistrează câţiva parametri asociaţi cu această acţiune, inclusiv adresa IP a clientului, documentul solicitat, codul de stare HTTP şi ora exactă la care s-a efectuat cererea. Listingul 2.5 arată exemple de intrări în fişierele jurnal. Capitolul 24, intitulat *Înregistrarea şi monitorizarea activităţii serverului Web*, vă va explica cum să specificaţi care parametri ai serverului să fie înregistraţi în fişierul jurnal.

Listingul 2.5. Exemple de intrări din fişierul access_log.

```
1: 127.0.0.1 - - [01/Sep/2003:09:43:37 -0700] "GET / HTTP/1.1" 200 1494
2: 127.0.0.1 - - [01/Sep/2003:09:43:40 -0700] "GET /manual/ HTTP/1.1" 200 10383
```

Fişierul error_log

Acest fişier include toate mesajele de eroare, mesajele de la pornire şi orice alt mesaj relevant apărut pe durata funcţionării serverului. Acesta este primul loc unde trebuie să căutaţi atunci când aveţi probleme cu serverul Apache. Listingul 2.6 vă arată exemple de intrări din fişierul error_log.

Listingul 2.6. Exemple de intrări din fişierul error_log.

```
1: [Sun Sep 01 09:42:59 2003] [notice] Parent: Created child process -2245
2: [Sun Sep 01 09:42:59 2003] [notice] Child -2242: Child process is running
3: [Sun Sep 01 09:42:59 2003] [notice] Child -2242: Acquired the start mutex.
4: [Sun Sep 01 09:42:59 2003] [notice] Child -2242: Starting 250 worker threads.
```

Fişiere suplimentare

Fişierul `httpd.pid` conţine identificatorii de proces ai serverului Apache care rulează într-un anumit moment. Puteţi folosi acest fişier pentru a trimite manual semnale serverului Apache, aşa cum este descris în secţiunea care urmează.

Fişierul `scoreboard`, prezent în cazul în care serverul Apache rulează pe platforma Linux/Unix, este folosit de modulele MPM care rulează în procese proprii pentru a comunica cu procesele lor copil.

În general, aceste fişiere nu trebuie să vă preocupe.

Comenzi ce pot fi trimise serverului Apache

Distribuţia serverului Apache conţine mai multe fişiere executabile. Această secţiune se referă doar la fişierul executabil al serverului şi la eventualele scripturi conexe. Capitolul 23, intitulat *Restricţionarea accesului la aplicaţiile dumneavoastră*, şi capitolul 26, *Reglarea performanţei serverului Apache şi folosirea host-urilor virtuale*, se ocupă şi de utilitarele suplimentare care sunt incluse în distribuţia standard a serverului Apache.

Fişierul executabil al serverului Apache

Fişierul executabil al serverului Apache se numeşte `httpd` pe platforma Linux/Unix şi `apache.exe` în cazul platformei Windows. Acest executabil acceptă mulţi parametri din linia de comandă, care sunt descrişi în tabelul 2.1. Puteţi obţine o listă completă a opţiunilor tastând, în orice moment, `/usr/local/apache2/bin/httpd -h` sub Linux/Unix sau `apache.exe -h` în linia de comandă sub Windows.

Tabelul 2.1. Opţiunile `httpd`.

Opţiune	Semnificaţie
`-D`	Vă permite să transmiteţi un parametru care poate fi folosit pentru procesarea unei secţiuni `<IfDefine>`.
`-l`	Enumeră modulele compilate.
`-v`	Afişează numărul versiunii şi data compilării serverului.
`-f`	Vă permite să transmiteţi calea către fişierul `httpd.conf` în cazul în care aceasta este diferită de cea implicită, de la momentul compilării.

În timp ce serverul Apache rulează, puteţi să folosiţi oricând comanda `kill` sub platforma Linux/Unix pentru a trimite semnale procesului părinte al serverului Apache. Semnalele oferă un mecanism de a trimite comenzi către un proces. Pentru a trimite un semnal, utilizaţi următoarea comandă:

```
# kill -SIGNAL pid
```

unde *pid* este identificatorul de proces, iar SIGNAL este unul dintre următoarele semnale:

- HUP — oprirea serverului;
- USR1 sau WINCH — repornire elegantă; semnalul depinde de sistemul de operare folosit;
- SIGHUP — repornire.

În cazul în care aţi operat modificări în fişierul de configurare şi doriţi ca ele să-şi facă efectul, trebuie să semnalaţi acest lucru serverului Apache. Puteţi face asta oprind şi repornind serverul, sau trimiţând semnalul de reiniţializare. Acest lucru instruieşte serverul să îşi reîncarce fişierele de configurare.

O repornire normală poate avea ca rezultat o pauză de moment în furnizarea serviciilor. O repornire elegantă reprezintă o abordare diferită. Toate procesele care servesc în acel moment un client vor continua să-şi îndeplinească misiunea, dar în momentul în care au terminat vor fi închise, locul lor fiind luat de noi procese care au deja încărcate noile configurări. Acest lucru permite operarea fără întrerupere a serverului, fără ca acesta să fie oprit vreo secundă.

Pe sistemul de operare Windows, puteţi trimite următoarele semnale procesului `apache.exe`:

- `apache.exe -k restart` — reporneşte serverul Apache;
- `apache.exe -k gracefull` — reporneşte elegant serverul Apache;
- `apache.exe -k stop` — opreşte imediat serverul Apache.

Puteţi accesa aceste comenzi folosind scurtăturile create de programul de instalare a serverului Apache în meniul Start. Dacă aţi instalat serverul Apache ca serviciu de sistem puteţi porni sau opri serverul Apache folosind interfaţa de administrare a serviciilor Windows: în Control Panel, selectaţi Administrative Tools şi daţi clic pe pictograma Services.

Scriptul de control al serverului Apache

Deşi este posibil să controlaţi serverul Apache pe platforma Linux/Unix folosind fişierul binar `httpd`, este recomandat să folosiţi scriptul de iniţializare `apachectl`. Pentru a folosi acest script tastaţi:

```
# /usr/local/apache2/bin/apachectl comanda
```

unde *comanda* înseamnă `stop`, `start`, `restart` sau `gracefull`. De asemenea, puteţi edita fişierul `apachectl` pentru a adăuga opţiuni suplimentare.

Unele distribuţii de sisteme de operare vă oferă diverse alte scripturi pentru controlarea serverului Apache. Citiţi documentaţia respectivei distribuţii înainte de a le folosi.

Prima pornire a serverului Apache

Înainte de a încerca inițializarea serverului Apache ar trebui să verificați existența unui set minimal de configurări în fișierul `httpd.conf`. Următoarele secțiuni descriu informațiile de bază despre configurarea serverului Apache și, de asemenea, vă învață cum să inițializați serverul.

Verificați-vă fișierul de configurare

Puteți edita fișierul `httpd.conf` cu editorul dumneavoastră preferat de text. Sub Linux/Unix, acest lucru înseamnă, probabil, folosirea utilitarelor `vi` sau `emacs`. Sub Windows, puteți folosi Notepad sau WordPad; trebuie, totuși, să vă amintiți de fiecare dată să salvați fișierele de configurare în format plain text (text neformatat), acesta fiind unicul format pe care serverul Apache îl înțelege.

Doi parametri trebuie configurați înainte de a încerca să rulați serverul Apache pentru prima oară și anume: numele serverului, pe de-o parte și, pe de altă parte, adresa IP și portul pe care acesta va asculta. Numele serverului este numele pe care serverul îl folosește atunci când se referă la el însuși (de exemplu, atunci când redirecționează cereri).

De obicei, serverul Apache își dă seama automat care este numele pe care trebuie să îl folosească, pe baza adresei IP a stației pe care a fost instalat, dar acest lucru nu se întâmplă întotdeauna. Dacă serverul are o intrare DNS, ar trebui să folosiți una dintre adresele IP în loc de nume. Dacă serverul nu este conectat la o rețea (s-ar putea să doriți să testați serverul Apache pe o stație care nu este conectată la o rețea), puteți folosi valoarea `127.0.0.1`, care este adresa de loopback. Valoarea implicită a portului pe care ascultă serverul este 80. S-ar putea să fie nevoie să schimbați această valoare dacă aveți deja un server Web care ascultă pe acest port, sau dacă nu aveți drepturi administrative — doar un utilizator cu drepturi administrative poate să asocieze porturi privilegiate (acele porturi cu numărul sub 1024) pe platformele Linux/Unix.

Puteți schimba atât adresa cât și portul folosind directiva Listen. Directiva Listen acceptă ca parametru fie un număr de port, fie o adresă și un port separate de două puncte (:). Dacă este specificat doar portul, serverul Apache va asculta pe acel port pe toate adresele disponibile pe acea stație. Dacă se specifică și o adresă IP, atunci serverul va asculta numai pe acea adresă și numai pe portul specificat. De exemplu, `Listen 80` instruiește serverul să asculte cereri pe toate adresele IP pe portul 80. Directiva `Listen 10.0.0.1:443` instruiește serverul Apache să asculte doar pe adresa `10.0.0.1` și pe portul `443`.

Directiva `ServerName` vă permite să definiți numele pe care serverul îl va raporta în orice adresă URL în care se vizează pe el însuși. Directiva acceptă un nume DNS și un port opțional, separate prin două puncte (:). Asigurați-vă că directiva ServerName are alocată o valoare validă. În caz contrar, serverul nu va funcționa corect; de exemplu, va efectua redirecționări incorecte.

Pe platforma Linux/Unix, puteţi folosi directivele `User` şi `Group` pentru a specifica sub ce utilizator şi grup va rula serverul. Utilizatorul `nobody` este o alegere bună pentru majoritatea platformelor. Totuşi, există unele probleme cu acest utilizator pe platforma HP-UX, aşa că, în acest caz, trebuie să creaţi un utilizator diferit, de exemplu `www`.

Iniţializarea serverului Apache

Pentru a porni serverul Apache pe platforma Linux/Unix, navigaţi până la directorul care conţine scriptul `apachectl` şi tastaţi următoarea comandă:

```
# /usr/local/apache2/bin/apachectl start
```

Pentru a iniţializa serverul Apache pe platforma Windows, daţi clic pe scurtătura Start Apache in Console, din secţiunea Control Apache Server, din grupul Apache HTTP Server 2.0.47, instalat în meniul Start. Dacă aţi instalat serverul Apache ca serviciu trebuie să porniţi serviciul.

Dacă totul decurge conform aşteptărilor, veţi putea accesa serverul Apache folosind un browser. Va apărea pagina iniţială de după instalare, aşa cum este ilustrată în figura 2.6. Dacă nu reuşiţi să iniţializaţi serverul sau dacă apare o pagină de eroare, vă rugăm să consultaţi secţiunea Depanare care urmează. Asiguraţi-vă că încercaţi să accesaţi serverul Apache pe unul dintre porturile pe care le-aţi specificat în directiva `Listen` — de obicei 80 sau 8080.

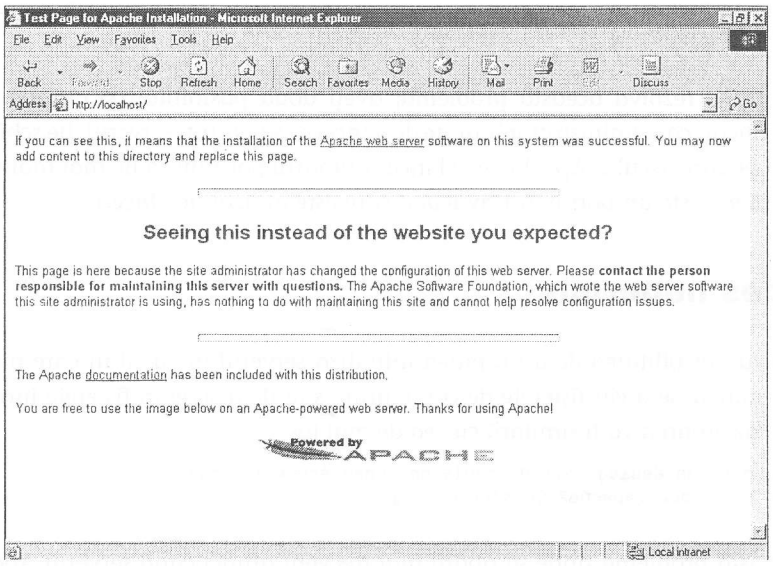


FIGURA 2.6. Pagina iniţială de instalare.

Depanare

În următoarele subsecţiuni vom descrie câteva dintre cele mai frecvente probleme pe care le-aţi putea întâlni la o primă încercare de iniţializare a serverului Apache.

Un server preexistent

Dacă aveţi deja un server care rulează pe acelaşi calculator şi care ascultă pe aceeaşi combinaţie de adresă IP şi port, serverul Apache nu va putea porni. În jurnalul de erori va fi adăugată o nouă intrare indicând faptul că Apache nu s-a putut ataşa de portul respectiv:

```
[crit] (48)Address already in use: make_sock: could not bind to
➥address 10.0.0.2:80
[alert] no listening sockets available, shutting down
```

Pentru a rezolva această problemă, trebuie să opriţi serverul care rulează deja sau să schimbaţi configuraţia serverului Apache pentru a asculta pe alt număr de port.

Nu aveți permisiunea să alocați portul

Veţi întâmpina o eroare în cazul în care nu aveţi drepturi administrative şi încercaţi să alocaţi unul dintre porturile privilegiate (cu numărul între 0 şi 1024):

```
[crit] (13)Permission denied: make_sock: could not bind to address 10.0.0.2:80
[alert] no listening sockets available, shutting down
```

Pentru a rezolva această problemă, aveţi două posibilităţi şi anume fie vă autentificaţi ca administrator înainte de a încerca să porniţi serverul, fie schimbaţi portul pe care ascultă Apache astfel încât numărul portului să fie mai mare decât 1024 (8080 este un port non-privilegiat care este utilizat de obicei).

Acces nepermis

Există posibilitatea de a nu putea iniţializa serverul în cazul în care nu aveţi permisiunea de a citi fişierele de configurare sau de a scrie în fişierele jurnal. În acest caz eroarea va fi similară cu cea de mai jos:

```
(13)Permission denied: httpd: could not open error log file
    ➥ /usr/local/apache2/logs/error_log.
```

Această problemă tinde să apară mai frecvent atunci când serverul Apache a fost compilat şi instalat de un alt utilizator decât cel care încearcă să îl ruleze.

Configurări de grup greşite

Puteţi să configuraţi serverul Apache să ruleze sub acreditarea unui anumit utilizator sau grup de utilizatori. Apache are un set de valori implicite în ceea ce priveşte utilizatorul şi grupul sub care rulează. Uneori valorile implicite pot fi invalide, caz în care serverul va întoarce un mesaj de eroare care conţine `setgid: unable to set group id`.

Pentru a rezolva această eroare pe platforma Linux/Unix, trebuie să schimbaţi valoarea asociată directivei Group din fişierul de configurare cu o valoare care să fie validă pentru sistemul dumneavoastră de operare. Verificaţi fişierul `/etc/groups` pentru o listă a grupurilor existente.

Rezumat

În acest capitol v-au fost explicate diferite moduri de instalare a serverul Web Apache 2.0 pe platformele Linux/Unix şi Windows. De asemenea, a fost acoperită atât problema instalării fişierelor binare, cât şi a instalării pornind de la codurile sursă şi s-au explicat opţiunile de bază pe care le puteţi alege la momentul compilării. În plus, aţi învăţat despre amplasarea fişierelor de configurare a serverului, precum şi despre sintaxa comenzilor disponibile pentru modificarea configuraţiei serverului Apache. Aţi învăţat şi despre cele două fişiere jurnal foarte importante, `access_log` şi `error_log`. În cele din urmă, aţi văzut cum se porneşte şi cum se opreşte serverul cu ajutorul scripturile de control sau folosind direct fişierul executabil pe platformele Linux/Unix şi Windows.

Întrebări şi răspunsuri

Î. *Cum pot să instalez o versiune nouă fără a folosi configurările vechi?*

R. Dacă trebuie să compilaţi o nouă versiune a serverului Apache folosind codurile sursă şi nu doriţi ca rezultatul acestui proces să fie afectat de configurările preexistente, o idee foarte bună este să rulaţi comanda `make clean`. Aceasta se va ocupa de ştergerea fişierelor binare preexistente, a oricăror fişiere obiect şi a tuturor celorlalte fişiere vechi.

Î. *De ce sunt utile fişierele de configurare per director?*

R. Deşi fişierele de configurare per director tind să afecteze vizibil performanţele sistemului, ele pot fi totuşi extrem de utile pentru cazul în care drepturile administrative trebuie delegate. Din moment ce aceste fişiere sunt citite şi interpretate de fiecare dată când este primită o solicitare, nu este nevoie să reporniţi serverul dacă vreunul dintre aceste fişiere se modifică în vreun fel.

Puteţi să le permiteţi utilizatorilor site-ului dumneavoastră să efectueze modificări de configurare fără a fi nevoie să le acordaţi privilegii administrative. În acest mod ei pot, de exemplu, să protejeze cu parolă anumite porţiuni ale site-urilor lor Web.

Î. *Ce se înţelege printr-o directivă* `ServerName` *validă?*

R. Sistemul DNS este folosit pentru a asocia adresele IP cu numele de domeniu. Valoarea directivei `ServerName` este returnată de fiecare dată când serverul generează o adresă URL. Dacă folosiţi un anumit nume de domeniu trebuie să fiţi siguri că sistemul dumneavoastră DNS a fost instruit să rezolve această adresă, pentru ca ea să fie disponibilă clienţilor care vă vizitează site-ul.

Atelier

Atelierul este conceput să anticipeze posibile întrebări, să vă ajute să recapitulaţi cele învăţate şi să începeţi să vă puneţi cunoştinţele în practică.

Test

1. Cum puteţi preciza calea în care doriţi să se instaleze serverul Apache?
2. Care este principala diferenţă între secţiunile `<Location>` şi `<Directory>`?
3. Care este diferenţa între o repornire normală şi o repornire elegantă?

Soluții

1. Utilizatorii sistemului Linux/Unix pot folosi opţiunea `--prefix` la momentul execuţiei scriptului `configure`. Dacă în respectiva cale există deja o instalare, fişierele de configurare vor fi păstrate, dar vor fi înlocuite toate fişierele binare. Pe platforma Windows, această cale poate fi stabilită cu ajutorul programului de instalare.
2. Secţiunea `Directory` se referă la obiectele de tip sistem de fişiere; directiva `Location` se referă la elementele precum cele din bara de adrese a unui browser Web (denumite şi URI).
3. Într-un proces normal de repornire serverul este oprit şi apoi repornit, lucru care duce la pierderea unor solicitări. O repornire elegantă permite proceselor copil ale serverului Apache să continue să servească solicitările curente până în momentul în care pot fi oprite şi înlocuite cu procese noi care au noile configurări încărcate.

Activități

1. Exersaţi diferitele metode de oprire şi repornire a serverului.
2. Efectuaţi câteva modificări de configurare precum asocierea unor porturi diferite sau schimbarea valorii directivei `ServerName`.

CAPITOLUL 3
Instalarea şi configurarea pachetului PHP

În următoarele trei capitole, ultimele dedicate instalării componentelor software veţi obţine, instala şi configura pachetul PHP şi veţi efectua câteva modificări elementare serverului Apache pe care îl aveţi deja instalat.

În acest capitol, veţi învăţa:

- să instalaţi PHP pentru serverul Apache sub Linux/Unix;
- să instalaţi PHP pentru serverul Apache sub Windows;
- să verificaţi instalarea pachetului PHP pe sistemul dumneavoastră;
- să obţineţi asistenţă tehnică atunci când lucrurile nu merg cum ar trebui;
- elementele de bază ale limbajului PHP.

Versiunea curentă şi versiunile viitoare de PHP

Instrucţiunile de instalare din acest capitol se referă la versiunea 4.3.3 de PHP, care, în momentul scrierii acestei cărţi, reprezenta versiunea curentă. Grupul PHP lansează versiuni minore (diferenţiate prin cea de-a treia cifră a numărului versiunii) pentru actualizări care sporesc securitatea sistemului sau rezolvă erori de programare. Versiunile minore nu sunt lansate pe baza unui calendar de apariţie — Grupul PHP lansează o versiune minoră atunci când îmbunătăţirile sau rezolvările erorilor sunt create şi testate corespunzător.

În momentul în care citiţi această carte, este posibil ca numărul versiunii minore să se fi schimbat, ajungând la 4.3.4 sau chiar mai departe. În acest caz, trebuie să citiţi lista modificărilor, aflată la adresa `http://www.php.net/ChangeLog-4.php` pentru a afla dacă au apărut modificări în ceea ce priveşte procesul de instalare şi configurare, subiectul principal al acestui capitol.

Deşi este puţin probabil ca instrucţiunile de instalare să se schimbe de la o versiune minoră la alta, trebuie să vă deprindeţi să verificaţi întotdeauna ce modificări au fost aduse pachetului software pe care îl instalaţi sau pe care îl întreţineţi. Dacă, între momentul scrierii acestei cărţi şi momentul în care citiţi aceste rânduri, a apărut o versiune minoră nouă, fără ca instrucţiunile de instalare să se fi schimbat, trebuie

pur şi simplu să înlocuiţi mintal numărul versiunii, ori de câte ori acesta apare în instrucţiunile de instalare şi în figurile ce le acompaniază, cu noul număr al versiunii.

Apropo

> Pentru instrucţiuni despre instalarea pachetului PHP de pe CD-ROM, vă rugăm să consultaţi anexa A, *Instalarea aplicaţiilor MySQL, Apache şi PHP aflate pe CD-ROM-ul ataşat cărţii.*

Compilarea pachetului PHP pentru Apache sub Linux/Unix

În această secţiune, vom discuta despre o modalitate de instalare a pachetului PHP pentru Apache pe sistemele care rulează Linux / Unix. Procesul este mai mult sau mai puţin asemănător pentru orice sistem de operare de tip Unix. Deşi s-ar putea să găsiţi versiuni executabile de PHP pentru sistemul dumneavoastră, beneficiaţi de mai mult control dacă vă compilaţi singuri aplicaţia PHP pe baza codului sursă.

Pentru a descărca fişierele distribuţiei de PHP, mergeţi la adresa `http://www.php.net/` şi urmaţi linkul către secţiunea Downloads. Descărcaţi cea mai recentă versiune a codului sursă — în acest exemplu am folosit versiunea 4.3.3. Distribuţia dumneavoastră va avea un nume de genul `php-`*`versiune`*`.tar.gz`, unde *versiune* este numărul celei mai recente versiuni. Această arhivă este un fişier comprimat de tip `tar`, aşa că va trebui să îl despachetaţi:

```
# gunzip < php-versiune.tar.gz | tar xvf -
```

Păstraţi fişierul descărcat într-un director rezervat pentru fişierele cu cod sursă, cum ar fi `/usr/src` sau `/usr/local/src`. După ce aţi despachetat distribuţia, ar trebui să intraţi în directorul în care aceasta este amplasată:

```
# cd php-versiune
```

În interiorul acestui director, veţi găsi un script denumit `configure`. Acest script acceptă informaţii suplimentare atunci când este rulat din linia de comandă. Aceste argumente ce trebuie scrise în linia de comandă controlează caracteristicile pe care le va avea instalarea dumneavoastră de PHP. În acest exemplu, am inclus opţiunile de bază pe care trebuie să le folosiţi pentru a instala PHP astfel încât să funcţioneze împreună cu Apache şi MySQL. Vom discuta câteva dintre opţiunile disponibile ale scriptului `configure` mai târziu în acest capitol, precum şi în restul cărţii pe măsură ce ele devin relevante.

```
# ./configure  --prefix=/usr/local/php --with-mysql \
--with-apxs2=/usr/local/apache2/bin/apxs
```

O dată ce scriptul `configure` a fost executat, va apărea din nou prompterul liniei de comandă după ce vor fi afişate câteva note informative din partea Grupului PHP:

```
+---------------------------------------------------------------------+
| License:                                                            |
| This software is subject to the PHP License, available in this      |
| distribution in the file LICENSE.  By continuing this installation |
| process, you are bound by the terms of this license agreement.      |
| If you do not agree with the terms of this license, you must abort  |
| the installation process at this point.                             |
+---------------------------------------------------------------------+
|                         *** NOTE ***                                |
|          The default for register_globals is now OFF!               |
|                                                                     |
| If your application relies on register_globals being ON, you        |
| should explicitly set it to on in your php.ini file.                |
| Note that you are strongly encouraged to read                       |
| http://www.php.net/manual/en/security.registerglobals.php           |
| about the implications of having register_globals set to on, and    |
| avoid using it if possible.                                         |
+---------------------------------------------------------------------+

Thank you for using PHP.
```

În continuare, daţi comanda `make`, urmată de comanda `make install`. Aceste comenzi ar trebui să finalizeze procesul de compilare şi instalare a pachetului PHP şi să fie urmate de afişarea prompterului de linie de comandă.

Va trebui să vă asiguraţi că două fişiere importante sunt copiate în directoarele corecte. Mai întâi, scrieţi următoarea comandă pentru a copia versiunea de distribuţie a fişierului `php.ini` în directorul corespunzător. Veţi învăţa mai multe despre fişierul `php.ini` mai târziu în acest capitol.

```
# cp php.ini-dist /usr/local/lib/php.ini
```

După aceasta, copiaţi fişierul-obiect partajat al PHP-ului în directorul corespunzător din calea de instalare a serverului Apache, în cazul în care nu a fost amplasat acolo de către procesul de instalare:

```
# cp libs/libphp4.so /usr/local/apache2/modules/
```

Acum ar trebui să puteţi configura şi rula serverul Apache, dar haideţi să discutăm câteva aspecte suplimentare despre opţiunile de configurare înainte de a ne îndrepta către secţiunea intitulată *Integrarea dintre PHP şi Apache sub Linux/Unix*.

Opţiuni suplimentare de configurare

Atunci când am rulat scriptul `configure`, am inclus câteva argumente din linia de comandă care au indus exprimarea câtorva caracteristici ale motorului de PHP. Scriptul `configure` ne poate oferi el însuşi o listă a opţiunilor disponibile, inclusiv cele pe care le-am folosit noi. Din directorul în care se găseşte distribuţia de PHP, tastaţi următoare comandă:

```
# ./configure --help
```

Rezultatul rulării acestei comenzi este o listă lungă, aşa că s-ar putea să doriţi să o salvaţi într-un fişier pentru a o citi în linişte:

```
# ./configure  --help  > configoptions.txt
```

Dacă descoperiţi funcţii suplimentare pe care doriţi să le adăugaţi după ce aţi instalat pachetul PHP, nu trebuie decât să efectuaţi din nou procesele de configurare şi compilare. Dacă faceţi acest lucru, va fi creată o nouă versiune a fişierului-obiect `libphp4.so` pe care va trebui să îl copiaţi în structura de directoare în care este instalat serverul Apache. După aceasta, trebuie doar să reporniţi serverul Apache pentru ca noul fişier să fie reîncărcat.

Integrarea dintre PHP şi Apache sub Linux/Unix

Pentru a vă asigura că PHP şi Apache se împacă unul cu celălalt, trebuie să verificaţi dacă există — şi, eventual, să adăugaţi — câteva linii în fişierul de configurare `httpd.conf`. Mai întâi, căutaţi o linie asemănătoare cu cea de mai jos:

```
LoadModule php4_module        modules/libphp4.so
```

Dacă această linie nu este prezentă, sau are un semn diez (#) la începutul său, trebuie să o adăugaţi sau să ştergeţi semnul #. Această linie îi spune serverului Apache să folosească fişierul-obiect partajat al pachetului PHP, fişier care a fost creat de procesul de compilare a surselor de PHP.

În continuare, căutaţi această secţiune:

```
#
# AddType allows you to add to or override the MIME configuration
# file mime.types for specific file types.
#
```

Adăugaţi următoarea linie:

```
AddType application/x-httpd-php .php .phtml .html
```

Acest lucru asigură faptul că motorul PHP va procesa fişierele care au extensia `.php`, `.phtml` şi `.html`. Extensiile pe care le selectaţi dumneavoastră ar putea fi altele şi aţi putea să doriţi, de exemplu, să adăugaţi între ele şi extensia `.php3`, pentru a asigura compatibilitatea cu scripturile foarte vechi pe care le-aţi putea avea.

Salvaţi fişierul `httpd.conf` şi reporniţi serverul Apache. Dacă vă uitaţi în fişierul `error_log`, ar trebui să vedeţi ceva asemănător cu linia de mai jos:

```
[Sun Sep 28 10:42:47 2002] [notice] Apache/2.0.47 (Unix) PHP/4.3.3 configured
```

PHP este acum parte integrantă din serverul de Web Apache. Dacă vreţi să învăţaţi cum să instalaţi PHP pe o platformă Windows, citiţi în continuare. Altminteri, puteţi trece direct la secţiunea *Verificarea instalării*, de la pagina 57.

Instalarea fişierelor PHP sub Windows

Spre deosebire de compilarea şi instalarea pachetului PHP sub Linux/Unix, instalarea omoloagă sub Windows nu implică decât descărcarea distribuţiei şi mutarea câtorva fişiere. Pentru a descărca fişierele de distribuţie, mergeţi la adresa `http://www.php.net/` şi urmaţi linkul către secţiunea Downloads. Descărcaţi cea mai recentă versiune a pachetului de tip `zip` (nu programul de instalare!) din secţiunea Windows Binaries — în acest exemplu, folosim versiunea 4.3.3. Distribuţia dumneavoastră va fi denumită `php-`*`versiune`*`.zip`, unde *`versiune`* este numărul celei mai recente versiuni.

O dată ce fişierul este descărcat pe calculatorul dumneavoastră, daţi-i dublu clic pentru a lansa programul dumneavoastră de dezarhivare. Distribuţia este arhivată în aşa fel încât include şi căile fiecărui fişier, aşa că puteţi extrage arhiva în directorul rădăcină al discului dumneavoastră, iar acolo va fi creat un director denumit `php-`*`versiune`*`-Win32` şi toate fişierele şi subdirectoarele vor fi amplasate în interiorul acestui director.

Acum că aveţi toate fişierele esenţiale ale distribuţiei de PHP, nu trebuie decât să mutaţi câteva dintre ele.

1. Identificaţi, în directorul în care s-a instalat pachetul PHP, un fişier intitulat `php.ini-dist` şi schimbaţi-i numele în `php.ini`.
2. Copiaţi fişierul `php.ini` în `C:\WINDOWS\` sau în directorul în care vă amplasaţi, de obicei, fişierele `*.ini`.
3. Copiaţi fişierul `php4ts.dll` în `C:\WINDOWS\SYSTEM\` sau în directorul în care vă amplasaţi, de obicei, fişierele `*.dll`.

Pentru ca o versiune simplă de PHP să funcţioneze împreună cu serverul Apache, trebuie să faceţi câteva modificări minore în fişierele de configurare ale acestui server.

Integrarea dintre PHP şi Apache sub Windows

Pentru a vă asigura că PHP şi Apache se împacă între ele, trebuie să adăugaţi câteva elemente în fişierul de configurare `httpd.conf` din directorul în care este instalat serverul Apache pe calculatorul dumneavoastră. Mai întâi, găsiţi o secţiune care arată la fel ca cea de mai jos:

```
# Example:
# LoadModule foo_module modules/mod_foo.so
#
LoadModule access_module modules/mod_access.so
LoadModule actions_module modules/mod_actions.so
LoadModule alias_module modules/mod_alias.so
LoadModule asis_module modules/mod_asis.so
LoadModule auth_module modules/mod_auth.so
#LoadModule auth_anon_module modules/mod_auth_anon.so
```

```
#LoadModule auth_dbm_module modules/mod_auth_dbm.so
#LoadModule auth_digest_module modules/mod_auth_digest.so
LoadModule autoindex_module modules/mod_autoindex.so
#LoadModule cern_meta_module modules/mod_cern_meta.so
LoadModule cgi_module modules/mod_cgi.so
#LoadModule dav_module modules/mod_dav.so
#LoadModule dav_fs_module modules/mod_dav_fs.so
LoadModule dir_module modules/mod_dir.so
LoadModule env_module modules/mod_env.so
#LoadModule expires_module modules/mod_expires.so
#LoadModule file_cache_module modules/mod_file_cache.so
#LoadModule headers_module modules/mod_headers.so
```

La sfârşitul secţiunii, adăugaţi următoarele:

```
LoadModule php4_module c:/php-versiune/sapi/php4apache2.dll
```

În continuare, treceţi această secţiune:

```
#
# AddType allows you to add to or override the MIME configuration
# file mime.types for specific file types.
#
```

Adăugaţi următoarea linie:

```
AddType application/x-httpd-php .php .phtml .html
```

Acest lucru ne asigură că motorul PHP va procesa fiecare fişier cu extensia `.php`, `.phtml` şi `.html`. Extensiile pe care le selectaţi dumneavoastră ar putea fi altele şi aţi putea să doriţi, de exemplu, să adăugaţi între ele şi extensia `.php3`, pentru a asigura compatibilitatea cu scripturile foarte vechi pe care le-aţi putea avea.

Salvaţi fişierul `httpd.conf` şi reporniţi serverul Apache. Serverul ar trebui să pornească fără să afişeze vreun avertisment. PHP este acum parte integrantă a serverului Web Apache.

Elementele de bază ale fişierului `php.ini`

După ce aţi compilat sau instalat pachetul PHP, încă mai puteţi să îi schimbaţi comportamentul modificând fişierul `php.ini`. Pe sistemele Unix, calea implicită a acestui fişier este `/usr/local/php/lib`, sau subdirectorul `lib` al directorului în care este instalat pachetul PHP. Pe sistemele Windows, acest fişier ar trebui să se găsească în directorul Windows.

Directivele din fişierul `php.ini` sunt de două feluri: valori şi flaguri. Directivele de tip valoare sunt formate din numele directivei şi o valoare, separate printr–un semn egal. Valorile posibile variază de la o directivă la alta. Directivele de tip `flag` sunt formate din numele directivei şi un indicator pozitiv sau negativ, separate printr-un semn egal.

Indicatorii pozitivi pot fi `1`, `On`, `Yes` şi `True`. Indicatorii negativi sunt `0`, `Off`, `No` şi `False`. Spaţiile albe sunt ignorate.

Puteţi să schimbaţi conţinutul fişierului `php.ini` în orice moment, dar, după ce faceţi acest lucru, va trebui să reporniţi serverul pentru ca modificările aduse să aibă efect. La un moment dat, citiţi fişierul `php.ini` singur pentru a vedea ce fel de elemente puteţi configura.

Verificarea instalării

Cea mai simplă modalitate prin care puteţi verifica instalarea pachetului PHP este să creaţi un script de test folosind funcţia `phpinfo()`. Această funcţie va returna o listă lungă de informaţii despre configurare. Deschideţi un editor de text şi scrieţi următoarea linie:

```
<?php phpinfo(); ?>
```

Salvaţi acest fişier sub numele `phpinfo.php` şi amplasaţi-l în directorul rădăcină al serverului dumneavoastră Web — subdirectorul `htdocs` din directorul în care este instalat serverul Apache. Accesaţi fişierul cu ajutorul browserului Web şi ar trebui să vedeţi ceva asemănător cu figurile 3.1 şi 3.2.

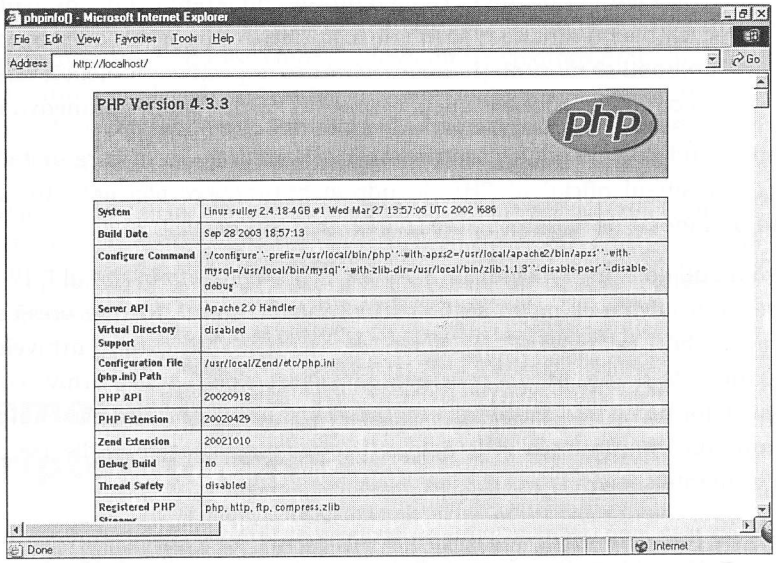


FIGURA 3.1. Rezultatele rulării funcţiei `phpinfo()` pe un sistem Linux/Unix.

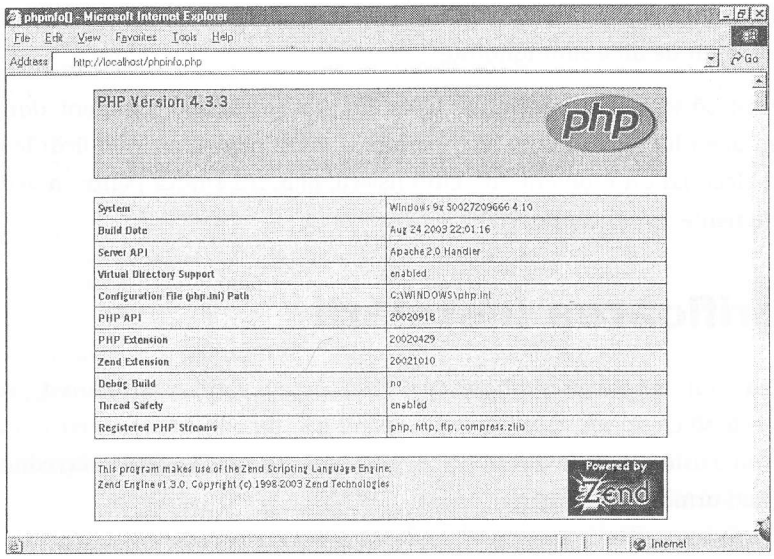


FIGURA 3.2. Rezultatele rulării funcției `phpinfo()` pe un sistem Windows.

Obținerea de asistență pentru instalare

Pe Internet puteți găsi întotdeauna foarte ușor asistență, mai ales pentru probleme legate de aplicațiile de tip *open source*. Așteptați, însă, un moment înainte de a trimite vreun mesaj prin care să cereți ajutorul. Indiferent cât de insolvabilă ar putea părea problema dumneavoastră de instalare, configurare sau programare, este foarte probabil că nu sunteți singurul care are respectiva problemă. Este chiar foarte posibil ca cineva să fi găsit deja răspunsul la întrebarea dumneavoastră.

Atunci când dați de greu, prima resursă informațională la care ar trebui să apelați este site-ul oficial al PHP, la adresa `http://www.php.net/` (în special manualul adnotat de la adresa `http://www.php.net/manual/`).

Dacă nu ați găsit încă răspunsul, nu uitați că puteți căuta în site-ul PHP. Sfatul de care aveți nevoie ar putea să fie ascuns într-un comunicat de presă sau în lista cu întrebări frecvente (FAQ). Puteți, de asemenea, să căutați arhivele listei de discuții de la adresa `http://www.php.net/search.php`. Aceste arhive conțin o cantitate enormă de informații reprezentând contribuțiile multor personalități de prim rang din comunitatea PHP. Petreceți ceva vreme încercând diverse combinații de cuvinte cheie.

Dacă sunteți, încă, ferm convins că problema dumneavoastră nu a fost abordată de nimeni, ați putea să aduceți un serviciu comunității PHP expunând-o în public. Puteți să vă înscrieți pe lista de discuții prin e-mail la adresa `http://www.php.net/support.php`. Deși aceste liste au debit infernal de informații, puteți învăța destule citind mesajele trimise de ceilalți membri. Dacă sunteți preocupat în mod serios de programarea în PHP, ar trebui să vă abonați cel puțin pentru

primirea rezumatelor zilnice de pe aceste liste. O dată abonat la listele care se potrivesc cu interesele dumneavoastră, puteţi să trimiteţi problema dumneavoastră pe aceste liste.

Atunci când trimiteţi o întrebare, este o idee bună să includeţi cât de multe informaţii puteţi (fără a scrie romane). Următoarele elemente sunt, de obicei, pertinente:

- sistemul dumneavoastră de operare;
- versiunea de PHP pe care o folosiţi sau încercaţi să o instalaţi;
- opţiunile de configurare pe care le alegeţi;
- orice text afişat de scriptul `configure` sau de comanda `make` înainte de a surveni eroarea;
- un exemplu cât de cât complet al codului care vă face probleme.

De ce atâtea precauţii înainte de a trimite întrebarea dumneavoastră pe o listă de discuţii? Înainte de toate, vă prinde bine să dobândiţi abilităţi de documentare. Un om care ştie să se documenteze poate, de obicei, să rezolve multe probleme rapid şi eficient. Trimiterea unei întrebări naive către o listă tehnică va fi răsplătită fie cu aşteptarea în van, fie cu câteva mesaje care vă vor indica arhivele în care ar fi trebuit, de fapt, să începeţi documentarea.

În al doilea rând, reţineţi că o listă de discuţie nu este un centru de asistenţă tehnică. Nimeni nu este plătit să vă răspundă la întrebări. În ciuda acestui fapt, aveţi privilegiul de a interacţiona cu o mulţime de oameni talentaţi şi cunoscători, printre care se găsesc chiar şi câţiva dintre creatorii limbajului PHP. Dacă puneţi o întrebare bună, răspunsul ar putea fi arhivat pentru a-i ajuta şi pe alţi programatori. Dacă puneţi o întrebarea la care s-a mai răspuns de câteva ori, nu faceţi altceva decât să contribuiţi la zgomotul de fond.

Acestea fiind spuse, nu trebuie, totuşi, să vă temeţi să puneţi întrebări pe o astfel de listă. Dezvoltatorii în PHP sunt oameni civilizaţi şi dornici să ajute, iar expunerea publică, în comunitate, a unei probleme ar putea să îi ajute să o rezolve şi pe alţi oameni.

Elementele de bază ale scripturilor PHP

Haideţi să trecem direct la scrierea unui script în PHP. Pentru a începe, deschideţi editorul dumneavoastră de text preferat. Aidoma documentelor HTML, fişierele PHP conţin text simplu. Puteţi să le creaţi cu orice editor de text, cum ar fi Notepad în Windows, Simple Text sau BBEedit în MacOS, sau vi şi Emacs în sistemele de operare de tip Unix. Cele mai cunoscute editoare de HTML pun la dispoziţie şi posibilitatea de a scrie scripturi PHP.

Ştiaţi că...

Keith Edmunds întreţine o listă a editoarelor care înlesnesc scrierea de PHP la adresa `http://phpeditors.linuxbackup.co.uk/`.

Tastaţi exemplul din listingul 3.1 şi salvaţi fişierul, denumindu-l `first.php`.

LISTINGUL 3.1. Un script simplu scris în PHP.

```
1: <?php
2:     echo "Hello Web!";
3: ?>
```

Dacă nu lucraţi direct pe maşina care va rula scriptul PHP, va trebui, probabil, să folosiţi un client de FTP, cum este WS-FTP pentru Windows sau Fetch pentru MacOS pentru a încărca fişierele pe server.

O dată ce documentul este în locul potrivit, ar trebui să îl puteţi accesa prin intermediul browserului. Dacă totul a mers bine, ar trebui să vedeţi rezultatul scriptului. Figura 3.3 arată rezultatul scriptului `first.php`.

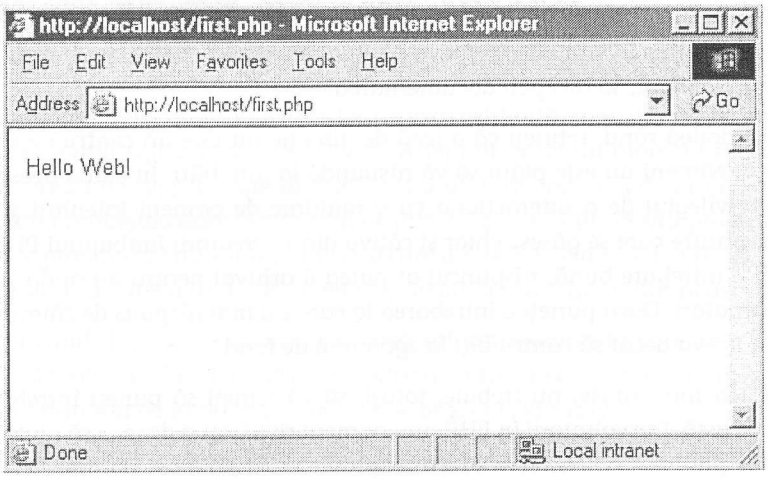


FIGURA 3.3. Am reuşit: rezultatul scriptului din listingul 3.1.

Deschiderea şi închiderea blocurilor cu instrucţiuni PHP

Atunci când scrieţi cod PHP, trebuie să informaţi motorul de PHP că doriţi să execute comenzile dumneavoastră. Dacă un faceţi asta, codul pe care îl scrieţi va fi considerat cod HTML şi va fi trimis către browser. Puteţi desemna codul ca fiind PHP folosind marcaje speciale care să indice începutul şi sfârşitul blocurilor de

cod PHP. Tabelul 3.1 arată patru astfel de marcaje folosite pentru delimitarea codului PHP.

TABELUL 3.1. Marcaje PHP de închidere şi de deschidere.

Stilul marcajului	Marcajul de deschidere	Marcajul de închidere
Marcaje standard	<?php	?>
Marcaje scurte	<?	?>
Marcaje în stil ASP	<%	%>
Marcaje de script	<SCRIPT LANGUAGE="php">	</SCRIPT>

Dintre marcajele din tabelul 3.1, numai marcajele standard şi cele de script funcţionează garantat în orice configuraţie. Marcajele scurte şi cele de tip ASP trebuie activate în `php.ini`.

Pentru a activa recunoaşterea marcajelor scurte, trebuie să vă asiguraţi că flagul `short_opne_tag` are valoarea `On` în `php.ini`:

```
short_open_tag  =  On;
```

Marcajele scurte sunt activate implicit, aşa că trebuie să editaţi `php.ini` numai dacă vreţi să le dezactivaţi.

Pentru a activa recunoaşterea marcajelor în stil ASP, trebuie să activaţi flagul `asp_tags`:

```
asp_tags = On;
```

După ce aţi editat fişierul `php.ini`, ar trebui să puteţi folosi toate cele patru stiluri în scripturile dumneavoastră. Folosirea unuia sau a altuia este, în general, o chestiune de preferinţă, deşi, dacă intenţionaţi să includeţi cod XML în scripturile dumneavoastră, ar trebui să dezactivaţi marcajele scurte (`<? ?>`) şi să lucraţi cu marcajele standard (`<?php ?>`).

Atenţie!

Secvenţa de caractere <? îi spune procesorului de XML să se aştepte să proceseze o instrucţiune şi, de aceea, apare frecvent în documentele XML. Dacă includeţi cod XML în scripturile dumneavoastră şi marcajele scurte sunt activate, motorul de PHP va confunda instrucţiunile de procesare XML cu marcaje de deschidere ale codului PHP. Dezactivaţi marcajele scurte dacă intenţionaţi să includeţi XML în documentele dumneavoastră.

Haideţi să parcurgem cele câteva modalităţi de scriere corectă a codului din listingul 3.1. Puteţi folosi oricare dintre marcajele de deschidere şi de închidere pe care le-aţi văzut:

```
<?
echo"Hello Web!";
?>

<?php
echo "Hello Web!";
?>

<%
echo "Hello Web!";
%>

<SCRIPT LANGUAGE="php">
echo "Hello Web!";
</SCRIPT>
```

Puteţi, de asemenea, să plasaţi linii unice de cod PHP între marcajele de deschidere şi închidere:

```
<? echo "Hello Web!"; ?>
```

Acum că ştiţi cum să definiţi un bloc de cod PHP, haideţi să ne uităm mai îndeaproape la codul din listingul 3.1.

Instrucţiunea `echo` şi funcţia `print()`

Simplu spus, instrucţiunea `echo` se foloseşte pentru afişarea de date. În cele mai multe cazuri, `echo` poate face cam orice să fie vizibil în browser. Puteţi, de asemenea, să folosiţi funcţia `print()` în locul instrucţiunii `echo`. Alegerea uneia dintre el este o chestiune de gust; atunci când vă uitaţi la scripturile altora, s-ar putea să le vedeţi folosite pe amândouă.

În timp ce `echo` este o instrucţiune, `print()` este o funcţie. O funcţie este o comandă care efectuează o acţiune, de obicei influenţată de datele pe care le primeşte. Datele trimise către o funcţie sunt aproape întotdeauna amplasate între paranteze, după numele funcţiei. În acest caz, aţi fi putut să îi trimiteţi funcţiei `print()` o colecţie sau un *şir de caractere — string*. Şirurile de caractere trebuie să fie întotdeauna plasate între ghilimele simple sau duble. Fie exemplul următor:

```
<?
print("Hello Web!");
?>
```

Singura linie de cod din listingul 3.1 se termină cu un semn punct şi virgulă. Acesta informează motorul de PHP că aţi terminat instrucţiunea.

O *instrucţiune* reprezintă o comandă trimisă către motorul de PHP. În sens larg, instrucţiunea reprezintă pentru PHP ceea ce propoziţia reprezintă pentru limba scrisă sau vorbită. O propoziţie se termină, în general, cu un punct — o instrucţiune trebuie să se termine cu punct şi virgulă. Excepţiile de la această regulă includ instrucţiunile în care sunt imbricate alte instrucţiuni şi instrucţiunile de la sfârşitul unui bloc de cod. De cele mai multe ori, totuşi, dacă nu încheiaţi o instrucţiune cu punct şi virgulă, motorul de PHP nu va mai înţelege nimic şi veţi provoca o eroare.

Combinarea codului HTML cu cod PHP

Scriptul din listingul 3.1 este PHP pur. Puteţi să încorporaţi acest cod într-un document HTML adăugând pur şi simplu cod HTML în jurul marcajelor de deschidere şi închidere ale blocului de cod PHP, aşa cum se vede în listingul 3.2.

LISTINGUL 3.2. Un script PHP înglobat în cod HTML.

```
 1: <html>
 2: <head>
 3: <title>Listing 3.2 A PHP script including HTML</title>
 4: </head>
 5: <body>
 6: <b>
 7: <?php
 8:     echo "hello world";
 9: ?>
10: </b>
11: </body>
12: </html>
```

După cum puteţi vedea, înglobarea codului PHP într-un document scris în cea mai mare parte în HTML implică numai scrierea codului efectiv. Motorul de PHP ignoră tot ce este între marcajele de deschidere şi închidere a blocului de cod PHP. Dacă aţi vizualiza listingul 3.2 într-un browser, aşa cum se vede în figura 3.4, aţi vedea şirul de caractere `hello world` scris cu litere aldine. Dacă aţi vizualiza sursa documentului, aşa cum se vede în figura 3.5, listingul ar arăta exact ca cel al unui document HTML normal.

Puteţi include mai multe blocuri de cod PHP într-un singur document, intercalând, acolo unde este nevoie, cod HTML. Deşi puteţi avea mai multe blocuri de cod într-un singur document, ele se combină formând un singur script. Orice variabile definite în primul bloc de cod vor fi, de obicei, disponibile şi în blocurile de cod următoare.

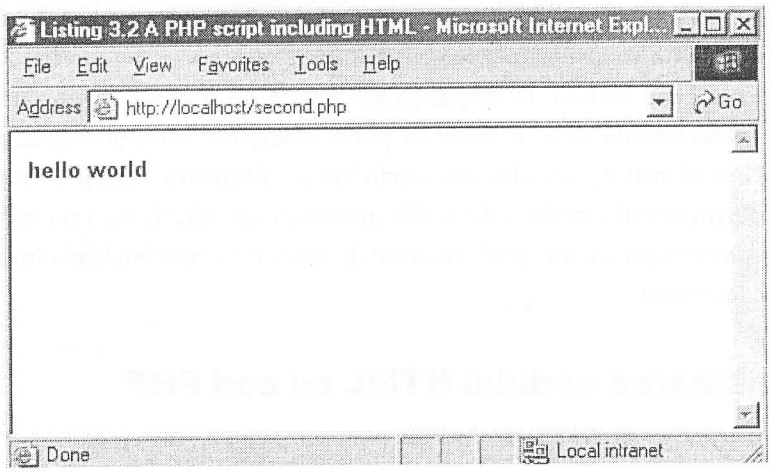


FIGURA 3.4. Rezultatul listingului 3.2, aşa cum se vede el într-un browser.

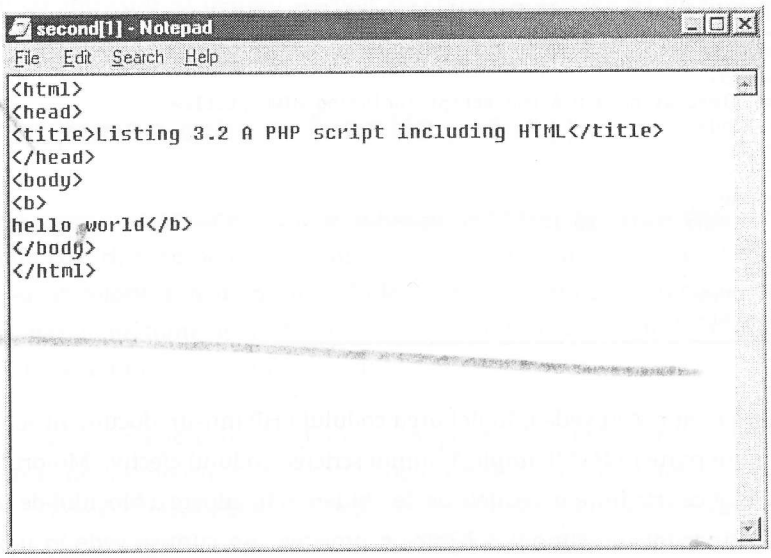


FIGURA 3.5. Rezultatul listingului 3.2 sub formă de cod HTML sursă.

Adăugarea de comentarii în codul dumneavoastră PHP

Codul pare foarte clar în momentul în care îl scrieţi, dar, atunci când încercaţi să îl modificaţi peste şase luni, vi se va părea imposibil de desluşit. Adăugarea de comentarii la codul dumneavoastră pe măsură ce îl scrieţi vă poate scăpa de timpul pierdut pentru înţelegerea sa ulterioară şi îi va ajuta pe alţi programatori să lucreze cu el.

Un comentariu reprezintă o porţiune de text din cadrul unui script care este ignorată de motorul de PHP. Comentariile pot fi folosite pentru a facilita înţelegerea unui script sau pentru a face adnotări.

Comentariile care se întind pe o singură linie încep cu două caractere / consecutive sau cu un singur caracter #. Motorul de PHP ignoră întreg textul aflat după aceste caractere, indiferent dacă se află la începutul sau la sfârşitul liniei:

```
// acesta este un comentariu
#  acesta este un alt comentariu
```

Comentariile ce se întind pe mai multe linii încep cu un caracter / urmat de un asterisc (*) şi se termină cu un asterisc urmat de un caracter /:

```
/*
acesta este un comentariu
si nimic din toate acestea
nu va fi procesat de catre
motorul de PHP
*/
```

Rezumat

În acest capitol, aţi învăţat cum să instalaţi şi să configuraţi pachetul PHP în vederea folosirii sale împreună cu serverul Apache, atât sub sistemele de operare Linux/Unix, cât şi sub Windows. Aţi învăţat diverse opţiuni ale scriptului `configure` cu ajutorul cărora puteţi schimba caracteristicile la momentul compilării în cazul sistemelor de tip Linux/Unix. Aţi învăţat despre fişierul `php.ini` şi despre cum puteţi să schimbaţi valorile directivelor conţinute de acesta. Aţi creat un script PHP simplu folosind un editor de text. Aţi examinat patru seturi de marcaje ce pot delimita blocurile de cod PHP. La final, aţi învăţat cum să folosiţi instrucţiunea `echo` sau funcţia `print()` pentru a trimite date către browser şi aţi pus cod HTML şi cod PHP la un loc în acelaşi script. În capitolul următor, veţi folosi aceste cunoştinţe pentru a verifica cele mai importante părţi componente ale limbajului PHP, printre care variabilele, tipurile de date şi operatorii.

Întrebări şi răspunsuri

Î. Aţi discutat despre instalarea pentru Linux/Unix sau Windows şi împreună cu serverul Web Apache. Asta înseamnă că materialul prezentat în această carte nu se aplică serverului şi sistemului de operare pe care le folosesc eu?

R. Ba da. Unul dintre punctele forte ale PHP-ului este că rulează pe mai multe platforme. Puteţi găsi instrucţiuni de instalare pentru diverse servere Web şi directive de configurare pentru operarea cu baze de date în Manualul de PHP. Deşi exemplele din această carte sunt axate în mod specific pe combinaţia PHP, MySQL şi Apache, nu este nevoie decât de modificări minore pentru a putea utiliza aceste scripturi astfel încât să funcţioneze folosind alt server de Web sau alt sistem de baze de date.

Î. Care sunt cele mai bune marcaje de deschidere şi închidere?

R. În mare măsură, depinde de preferinţă. În scopul portabilităţii, cele mai sigure şi cele mai bune marcaje sunt, probabil, cele standard (`<?php ?>`). Marcajele scurte sunt activate implicit şi au avantajul de a ocupa mai puţin spaţiu, dar, având în vedere popularitatea crescândă a limbajului XML, este mai bine să le evitaţi.

Î. Ce editoare de text ar trebui evitate pentru crearea scripturilor PHP?

R. Nu folosiţi editoare care formatează textul pentru imprimare (cum ar fi Word, de exemplu). Chiar dacă salvaţi fişierele create cu ajutorul unor astfel de programe în format text simplu, este posibil ca anumite caractere ascunse să se strecoare în codul dumneavoastră.

Î. Când ar trebui să adaug comentarii în codul meu?

R. Încă o dată, aceasta este o chestiune de preferinţe. Unele scripturi scurte ar putea fi uşor de înţeles fără ajutorul comentariilor, chiar şi după multă vreme. În cazul scripturilor cât de cât lungi sau complexe, ar trebui întotdeauna să vă comentaţi codul.

Atelier

Atelierul este conceput să anticipeze posibile întrebări, să vă ajute să recapitulaţi cele învăţate şi să începeţi să vă puneţi cunoştinţele în practică.

Test

1. Lucrând pe un sistem Linux/Unix, cum aţi putea să obţineţi lista opţiunilor de configurare (opţiunile pe care le puteţi pasa scriptului `configure` din distribuţia dumneavoastră de PHP)?

2. Ce linie ar trebui să adăugaţi în fişierul de configurare al serverului Apache pentru a vă asigura că extensia `.php` este recunoscută?
3. Cum se numeşte fişierul de configurare al pachetului PHP?
4. Poate o persoană care vizitează site-ul dumneavoastră Web să vadă codul sursă al scripturilor dumneavoastră atunci când PHP este instalat corect?

Soluții

1. Puteţi obţine liste opţiunilor de configurare apelând scriptul `configure` din distribuţia dumneavoastră de PHP şi pasându-i argumentul `--help`:

   ```
   ./configure --help
   ```

2. Linia:

   ```
   AddType application/x-httpd-php .php
   ```

   asigură recunoaşterea fişierelor cu extenisa `.php`.
3. Fişierul de configurare al PHP-ului se numeşte `php.ini`.
4. Nu, utilizatorul va vedea numai rezultatul scriptului dumneavoastră.

Activități

1. Instalaţi pachetul PHP pe sistemul dumneavoastră. Dacă acesta este deja instalat, treceţi în revistă fişierul `php.ini` şi verificaţi configuraţia.
2. Familiarizaţi-vă cu procesul de creare, încărcare pe server şi execuţie a scripturilor PHP. În special, creaţi-vă propriul script *„hello world"*. Adăugaţi-i cod HTML, precum şi blocuri suplimentare de cod PHP. Experimentaţi cu diverse marcaje de delimitare a codului PHP. Care dintre ele sunt activate în configuraţia dumneavoastră? Uitaţi-vă la fişierul `php.ini` pentru a vă confirma descoperirile. Nu uitaţi să adăugaţi comentarii codului dumneavoastră.

PARTEA a II-a

Structura limbajului PHP

Capitolul 4. Elementele constitutive ale limbajului PHP 71

Capitolul 5. Funcții de control al execuției codului în PHP 93

Capitolul 6. Folosirea funcțiilor 115

Capitolul 7. Lucrul cu vectorii şi obiectele 135

CAPITOLUL 4
Elementele constitutive ale limbajului PHP

În acest capitol, veţi lucra cu câteva dintre piesele componente ale limbajului PHP. În cazul în care sunteţi novice în ale programării, s-ar putea să vă simţiţi copleşit pe alocuri, dar nu vă speriaţi — puteţi să consultaţi din nou acest capitol în orice moment. Concentraţi-vă pe înţelegerea conceptelor şi nu pe memorizarea caracteristicilor discutate.

Dacă sunteţi deja un programator cu experienţă, ar trebui, cel puţin, să răsfoiţi acest capitol, deoarece discută câteva caracteristici specifice limbajului PHP.

În acest capitol, veţi învăţa:

- despre variabile — ce sunt ele, de ce avem nevoie să le folosim şi cum anume le folosim;
- cum să definiţi şi să accesaţi variabilele;
- despre tipurile de date;
- despre câţiva operatori mai frecvent utilizaţi;
- cum să folosiţi operatorii pentru a crea expresii;
- cum să definiţi şi să utilizaţi constante.

Variabilele

O variabilă este un container special pe care îl puteţi defini şi care, apoi, va putea „păstra" o valoare, cum ar fi un număr, un şir de caractere, un obiect, un vector sau o valoare booleană. Variabilele sunt elemente fundamentale ale programării. Fără variabile, am fi forţaţi să codificăm static toate valorile din scripturile noastre. Adunând două numere şi afişând rezultatul, puteţi obţine ceva folositor:

```
echo (2 + 4);
```

Totuşi, această bucăţică de cod nu le va fi folositoare decât oamenilor care vor să ştie rezultatul adunării lui 2 cu 4. Pentru a scăpa de acest dezavantaj, aţi putea scrie un script care să găsească suma altui set de numere, să zicem 3 şi 5. Evident, această abordare a programării este destul de absurdă, iar acesta este momentul în care variabilele intră în joc.

Variabilele ne permit să creăm şabloane pentru diverse operaţiuni (adunarea a două numere, de exemplu), fără să ne intereseze valorile efective implicate. Valorile vor fi primite de către variabile atunci când scriptul este rulat, de la utilizator, prin intermediul unei interogări asupra unei baze de date sau din rezultatul unei alte acţiuni efectuate mai devreme în cadrul scriptului, în funcţie de necesităţi. Cu alte cuvinte, variabilele sunt folosite ori de câte ori este probabil ca scriptul dumneavoastră să se schimbe — fie în cursul execuţiei sale, fie atunci când îi sunt pasate date de către un alt script.

O variabilă constă dintr-un nume ales de dumneavoastră, precedat de semnul dolar ($). Numele variabilelor pot include litere, cifre şi caracterul de subliniere (_), dar nu pot include spaţii. Numele variabilelor trebuie să înceapă cu o literă sau cu caracterul de subliniere. Următoarea listă conţine câteva nume legale de variabile:

```
$a;
$un_nume_lung_de_variabila;
$_2453;
$sleepyZZZZ;
```

Apropo

Numele variabilelor dumneavoastră trebuie să fie în acelaşi timp inteligibile şi coerente din punct de vedere stilistic. De exemplu, dacă scriptul lucrează cu nume de utilizatori şi parole, nu creaţi o variabilă denumită $n pentru nume şi $p pentru parolă — acestea nu sunt nume inteligibile. Dacă vă uitaţi la un astfel de script după câteva săptămâni de la scrierea lui, s-ar putea să credeţi că $n este variabila pentru „număr", nu pentru „nume" şi că $p înseamnă „pagină" în loc de „parolă".

Caracterul punct şi virgulă (;) — cunoscut şi sub numele de *terminator de instrucţiune* — este folosit pentru a încheia o instrucţiune PHP. Caracterele punct şi virgulă din fragmentul anterior de cod nu fac parte din numele variabilelor, dar sunt folosite pentru a încheia fiecare dintre liniile pe care sunt definite variabilele.

După cum puteţi vedea, aveţi la dispoziţie o gamă largă de opţiuni atunci când daţi nume variabilelor. Pentru a declara o variabilă, nu trebuie decât să o includeţi în scriptul dumneavoastră. Atunci când declaraţi o variabilă, de obicei îi şi atribuiţi o valoare în cadrul aceleiaşi instrucţiuni, ca mai jos:

```
$num1 = 8;
$num2 = 23;
```

Liniile de cod precedente declară două variabile, folosind operatorul de atribuire (=) pentru a le da valori. Veţi învăţa despre atribuire în mai mare detaliu în cadrul secţiunii „Operatori şi expresii" de mai jos, în acest capitol. După ce le atribuiţi valori variabilelor dumneavoastră, le puteţi trata ca şi când ar fi valorile în sine. Cu alte cuvinte, linia

```
echo $num1;
```

este echivalentă cu:

```
echo 8;
```

atâta vreme cât variabila `$num` are valoarea 8.

Variabile globale şi superglobale

Pe lângă regulile de numire a variabilelor, există şi reguli privind disponibilitatea variabilelor. În general, valoarea atribuită unei variabile este prezentă doar în interiorul funcţiei sau scriptului în cadrul căruia se găseşte. De exemplu, dacă aveţi un script, `scriptA.php`, care are o variabilă cu numele `$nume` şi valoarea `joe` şi doriţi să mai creaţi un script, `scritpB.php`, care foloseşte tot o variabilă `$nume`, puteţi să îi atribuiţi acesteia din urmă valoarea `jane` fără a afecta scriptul `scriptA.php`. Valoarea variabilei `$nume` este locală la nivelul fiecărui script şi valorile atribuite sunt independente una de cealaltă.

Cu toate acestea, puteţi, de asemenea, să definiţi variabile `$nume` drept *variabilă globală* într-un script sau într-o funcţie. În acest caz `scriptA.php` şi `scriptB.php` vor fi conectate, existând o singură valoare pentru variabila `$nume` care, acum, este partajată. Exemple cu variabile globale şi cu domeniile lor de definire vor fi discutate mai în detaliu în capitolul 6, intitulat *Folosirea funcţiilor*.

Alături de variabilele globale create de dumneavoastră, PHP mai are şi câteva variabile predefinite care se numesc *variabile superglobale*. Aceste variabile sunt prezente întotdeauna, iar valorile lor sunt disponibile pentru toate scripturile dumneavoastră. Fiecare dintre variabilele superglobale de mai jos sunt, de fapt, vectori care conţin alte variabile:

- `$_GET` conţine orice variabile care îi sunt transmise scriptului prin metoda `GET`;
- `$_POST` conţine orice variabile care îi sunt transmise scriptului prin metoda `POST`;
- `$_COOKIE` conţine orice variabile care îi sunt transmise scriptului prin intermediul cookie-urilor;
- `$_FILES` conţine orice variabile care îi sunt transmise scriptului prin intermediul procedurii de încărcare a fişierelor pe server;
- `$_SERVER` conţine informaţii cum ar fi anteturile, căile către fişiere şi amplasarea fizică a scripturilor;
- `$_ENV` conţine orice variabile care din mediul de lucru al serverului şi care sunt accesibile scriptului;
- `$_REQUEST` conţine orice variabile care îi sunt transmise scriptului prin intermediul mecanismului de introducere a datelor de către utilizator;
- `$_SESSION` conţine orice variabile care îi sunt înregistrate într-o sesiune de utilizator.

Apropo

Dacă folosiți o versiune de PHP care este mai veche de 4.1.x şi nu puteți să actualizați pachetul PHP la o versiune mai nouă, trebuie să ajustați numele variabilelor atunci când lucrați cu scripturile prezentate în carte. Numele vechi ale acestora sunt `$HTTP_GET_VARS` (pentru `$_GET`), `$HTTP_POST_VARS` (pentru `$_POST`), `$HTTP_COOKIE_VARS` (pentru `$_COOKIE`), `$HTTP_POST_FILES` (pentru `$_FILES`), `$HTTP_ENV_VARS` (pentru `$_ENV`) şi `$HTTP_SESSION_VARS` (pentru `$_SESSION`). În versiunile vechi, aceste variabile sunt vectori simpli, nu superglobali şi trebuie definiți ca atare.

Exemplele din această carte vor folosi vectorii superglobali ori de câte ori va fi posibil. Folosirea acestora în cadrul scripturilor dumneavoastră este importantă în vederea creării de aplicații securizate, deoarece reduce riscul injectării de cod în scripturile dumneavoastră. Scriind scripturile în aşa fel încât să nu accepte decât datele pe care le doriţi, în modalitatea pe care o doriți, puteți elimina câteva dintre problemele ridicate de scripturile scrise neglijent.

Tipuri de date

Tipuri diferite de date ocupă volume diferite de memorie şi pot fi tratate diferit atunci când sunt manipulate de către un script. De aceea, unele limbaje de programare îi cer programatorului să declare explicit şi dinainte tipul de date pe care le va conţine o variabilă. Din contră, limbajul PHP este foarte libertin în ceea ce priveşte tipurile de date, acesta fiind stabilit atunci când variabilei i se atribuie o valoare. Acest lucru este, în acelaşi timp, o binecuvântare şi un blestem. Pe de-o parte, înseamnă că variabilele pot fi utilizate în mod flexibil, o aceeaşi variabilă putând conţine un şir de caractere la un moment dat şi, în alt moment, un număr întreg. Pe de altă parte, acest lucru poate cauza probleme în cazul scripturilor mai mari dacă vă aşteptaţi ca o variabilă să conţină un tip de date, când, de fapt, ea conţine cu totul şi cu totul altceva. De exemplu, să considerăm că aţi creat un script care este conceput să opereze cu o variabilă de tip vector. Dacă variabila respectivă conţine de fapt un număr, se pot produce erori atunci când codul încearcă să efectueze operaţiuni specifice vectorilor asupra acelei variabile.

Tabelul 4.1 arată tipurile standard de date care sunt disponibile în PHP.

TABELUL 4.1. Tipurile standard de date.

Tip	Exemplu	Descriere
`Integer`	`5`	Un număr întreg.
`Double`	`3.234`	Un număr real în virgulă mobilă.
`String`	`"hello"`	O colecție de caractere.
`Boolean`	`true`	Una dintre valorile speciale `true` sau `false`.
`Object`		O instanță a unei clase.
`Array`		Un set ordonat de perechi cheie - valoare.

PHP mai pune, de asemenea, la dispoziţie două tipuri speciale de date, enumerate în tabelul 4.2.

TABELUL 4.2. Tipurile speciale de date.

Tip	Descriere
`Resource`	O referinţă către o resursă pusă la dispoziţie de o terţă parte (o bază de date, de exemplu).
`NULL`	O variabilă neiniţializată.

Datele de tip `Resource` sunt, de obicei, returnate de funcţiile care operează cu fişiere sau cu aplicaţii externe. Tipul de date `NULL` este rezervat pentru variabilele care nu au fost iniţializate (adică variabile cărora nu le-a fost încă atribuită nici o valoare).

Puteţi folosi funcţia PHP predefinită `gettype()` pentru a verifica tipul oricărei variabile. Dacă amplasaţi numele variabilei între parantezele apelului către această funcţie, `gettype()` returnează un şir de caractere ce reprezintă tipul de date respectiv. Listingul 4.1 atribuie patru tipuri de date diferite unei aceleiaşi variabile, verificând rezultatul de fiecare dată cu ajutorul funcţiei `gettype()`. Comentariile din cod spun ce face scriptul în fiecare moment.

Apropo

Puteţi citi mai multe despre apelarea funcţiilor în capitolul 6, intitulat *Folosirea funcţiilor*.

LISTINGUL 4.1. Verificarea tipului unei variabile.

```
<html>
<head>
<title>Listing 4.1 Testing the type of a variable</title>
</head>
<body>
<?php
$testing; // declarare fara atribuire
echo gettype($testing); // null
echo "<br>";
$testing = 5;
echo gettype($testing); // integer
echo "<br>";
$testing = "five";
echo gettype($testing); // string
echo "<br>";
$testing = 5.0;
echo gettype($testing); // double
echo "<br>";
$testing = true;
echo gettype($testing); // boolean
echo "<br>";
?>
</body>
</html>
```

Puneţi aceste linii într-un fişier text denumit `gettype.php` şi amplasaţi fişierul în directorul rădăcină al serverului dumneavoastră Web. Atunci când accesaţi acest script prin intermediul browserului, el va produce următoarele rezultate:

```
NULL
integer
string
double
boolean
```

Când declarăm variabila `$testing` pe linia 7, nu îi atribuim nici o valoare, aşa că prima oară când folosim funcţia `gettype()` pentru a testa variabila, pe linia 8, obţinem şirul de caractere `NULL`. După aceasta, îi atribuim variabilei `$testing` valori folosind semnul = înainte de a o pasa, de fiecare dată, funcţiei `$gettype()`. Valoarea de tip `integer` pe care o atribuim variabilei `$testing` pe linia 10 este un număr real întreg. Simplu spus, este vorba despre un număr care nu are zecimale. Valoarea de tip `string` pe care o atribuim variabilei `$testing` pe linia 13 este o colecţie de caractere. Atunci când lucraţi cu şiruri de caractere în scripturile dumneavoastră, ele trebuie întotdeauna să fie puse în ghilimele duble sau simple (" sau '). Valoarea de tip `double` pe care o atribuim variabilei `$testing` pe linia 16 este un număr real în virgulă mobilă (adică un număr care conţine şi zecimale). Valoarea de tip `boolean` pe care o atribuim variabilei `$testing` pe linia 19 poate avea două valori speciale: `true` (adevărat) sau `false` (fals).

Schimbarea tipului prin folosirea funcţiei `settype()`

PHP ne pune la dispoziţie funcţia `settype()` pentru a schimba tipul unei variabile. Pentru a folosi `settype()` trebuie să puneţi numele variabilei al cărei tip vreţi să îl schimbaţi, precum şi noul tip pe care trebuie să îl capete aceasta, între parantezele funcţiei `settype()`, separate prin virgulă. Listingul 4.2 converteşte valoarea 3.14 (de tip `double`) în fiecare dintre cele patru tipuri de date asupra cărora ne concentrăm în acest capitol.

LISTINGUL 4.2. Schimbarea tipului unei variabile prin folosirea funcţiei `settype()`.

```
 1: <html>
 2: <head>
 3: <title>Listing 4.2 Changing the type of a variable with settype()</title>
 4: </head>
 5: <body>
 6: <?php
 7: $undecided = 3.14;
 8: echo gettype($undecided); // double
 9: echo " is $undecided<br>";  // 3.14
10: settype($undecided, 'string');
11: echo gettype($undecided); // string
12: echo " is $undecided<br>";  // 3.14
13: settype($undecided, 'integer');
14: echo gettype($undecided); // integer
15: echo " is $undecided<br>";  // 3
16: settype($undecided, 'double');
```

LISTINGUL 4.2. *(continuare)*

```
17: echo gettype($undecided); // double
18: echo " is $undecided<br>";  // 3
19: settype($undecided, 'boolean');
20: echo gettype($undecided); // boolean
21: echo " is $undecided<br>";  // 1
22: ?>
23: </body>
24: </html>
```

În fiecare caz, folosim funcţia `gettype()` pentru a confirma faptul că schimbarea de tip a funcţionat şi, apoi, afişăm valoarea variabilei `$undecided` în browser. Când convertim şirul de caractere `"3.14"` în număr întreg, pe lina 13, orice informaţie care se află dincolo de punctul care marchează zecimalele este pierdută pentru totdeauna. Acesta este motivul pentru care variabila `$undecided` conţine valoarea `3.0` chiar şi după ce o transformăm din nou în număr de tip double, pe linia 16. În fine, pe lina 19, schimbăm tipul variabilei `$undecided` în `boolean`. Orice număr care nu este `0` devine, atunci când este convertit în boolean, `true`. Atunci când este afişată o valoarea de tip boolean în PHP, `true` este reprezentat sub forma `1`, iar false sub forma unui şir vid, aşa că, pe linia 21, variabila `$undecided` este afişată ca `1`.

Puneţi aceste linii într-un fişier cu numele `settype.php` şi amplasaţi fişierul în directorul rădăcină al serverului dumneavoastră Web. Atunci când accesaţi acest script prin intermediul browserului, el produce următoarele rezultate:

```
double is 3.14
string is 3.14
integer is 3
double is 3
boolean is 1
```

Schimbarea tipului prin casting (conversia valorii variabilei)

Scriind numele unui tip de date, în paranteze, înaintea numelui unei variabile, creaţi o copie a valorii respectivei variabile convertită în tipul de date specificat.

Diferenţa principală între folosirea funcţiei `settype()` şi casting constă în aceea că, prin casting, obţineţi o copie a valorii variabilei, lăsând variabila originală intactă. Listingul 4.3 ilustrează acest lucru.

LISTINGUL 4.3. Castingul unei variabile.

```
1: <html>
2: <head>
3: <title>Listing 4.3 Casting a variable</title>
4: </head>
5: <body>
6: <?php
7: $undecided = 3.14;
8: $holder = (double) $undecided;
```

LISTINGUL 4.3. *(continuare)*

```
 9: echo gettype($holder) ; // double
10: echo " is $holder<br>";   // 3.14
11: $holder = (string) $undecided;
12: echo gettype($holder);  // string
13: echo " is $holder<br>";   // 3.14
14: $holder = (integer) $undecided;
15: echo gettype($holder);  // integer
16: echo " is $holder<br>";   // 3
17: $holder = (double) $undecided;
18: echo gettype($holder);  // double
19: echo " is $holder<br>";   // 3.14
20: $holder = (boolean) $undecided;
21: echo gettype($holder);  // boolean
22: echo " is $holder<br>";   // 1
23: echo "<hr>";
24: echo "original variable type: ";
25: echo gettype($undecided); // double
26: ?>
27: </body>
28: </html>
```

Nu am modificat efectiv la nici un moment tipul variabilei `$undecided`, care rămâne `double` pe tot parcursul scriptului. Acest lucru este ilustrat pe linia 25, când folosim funcţia `gettype()` pentru a afişa tipul lui `$undecided`.

De fapt, folosind castingul asupra variabilei `$undecided`, am creat o copie a valorii sale care, apoi, a fost convertită în tipul de date specificat. Această nouă valoare este stocată în variabila `$holder`, prima oară pe linia 8 şi, de asemenea, pe liniile 11, 14, 17 şi 20. Deoarece lucrăm cu o copie a valorii variabilei `$undecided`, nu pierdem niciodată informaţiile conţinute de aceasta, aşa cum s-a întâmplat pe liniile 13 şi 19 ale listingului 14.2.

Puneţi aceste linii într-un fişier text denumit `testcast.php` şi amplasaţi acest fişier în directorul rădăcină al serverului dumneavoastră Web. Atunci când accesaţi acest script în browser, el produce următoarele rezultate:

```
double is 3.14
string is 3.14
integer is 3
double is 3.14
boolean is 1
original variable type: double
```

Acum că putem schimba tipul de date al unei variabile folosind fie `settype()`, fie castingul, ar trebui să ne gândim la ce ne-ar putea ajuta aşa ceva. Cu siguranţă că nu veţi folosi aceste proceduri prea des, deoarece PHP efectuează conversiile de tip în mod automat pentru dumneavoastră atunci când acest lucru se impune. Cu toate acestea, conversia automată este temporară şi s-ar putea să doriţi să faceţi în aşa fel ca o variabilă să îşi păstreze în mod persistent tipul de date.

Numerele pe care un utilizator le tastează într-un formular HTML vor fi preluate de către scriptul dumneavoastră sub forma unor şiruri de caractere. Dacă încercaţi să adunaţi două şiruri de caractere ce conţin numere, PHP vă va sări în ajutor convertind şirurile de caractere în numere atunci când are loc adunarea. Astfel:

```
"30cm" + "40cm"
```

va avea drept rezultat numărul întreg 70. Atunci când converteşte şiruri de caractere, PHP ignoră caracterele non-numerice. S-ar putea, totuşi, să doriţi să curăţaţi datele introduse de utilizator chiar dumneavoastră. Imaginaţi-vă că utilizatorului i s-a cerut să introducă un număr. Putem simula acest lucru declarând o variabilă şi atribuindu-i o valoare:

```
$test = "30cm";
```

După cum puteţi vedea, utilizatorul a adăugat şi unitatea de măsură. Ne putem asigura că datele introduse de utilizator sunt curate convertindu-le prin casting în tipul de date `integer`:

```
$newtest = (integer)$test;
echo "Your imaginary box has a width of $test centimeters";
```

De ce să verificăm tipul de date?

De ce ar putea fi util să ştim tipul de date al unei variabile? De multe ori, în programare, ne găsim în situaţia în care datele cu care lucrăm provin din alte surse. În capitolul 6, de exemplu, veţi învăţa cum să creaţi funcţii în scripturile dumneavoastră. Funcţiile pot accepta informaţii din partea codului care le apelează sub forma argumentelor. Pentru ca funcţia să poată să lucreze cu datele care îi sunt furnizate, de multe ori, este o idee bună să verificăm mai întâi că valorile respective sunt de tipul corect. O funcţie care aşteaptă o valoare de tip `resource`, de exemplu, nu va opera corespunzător atunci când îi este pasat un şir de caractere.

Operatori şi expresii

Cu ceea ce aţi învăţat până acum, puteţi să atribuiţi date variabilelor. Puteţi chiar şi să aflaţi şi să schimbaţi tipul de date al unei variabile. Un limbaj de programare, însă, nu ar fi prea util dacă nu aţi putea manipula datele pe care le-aţi stocat. Operatorii sunt simboluri care permit folosirea uneia sau a mai multor valori pentru producerea unei noi valori. Valoarea cu care operează un operator se numeşte operand.

Apropo

Un *operator* este un simbol sau o serie de simboluri care, atunci când sunt folosite în conjuncţie cu valori, efectuează o acţiune şi, de obicei, produce o valoare nouă.

Un *operand* este o valoarea folosită în conjuncţie cu un operator. De obicei, se folosesc doi operanzi şi un operator.

Haideţi să combinăm doi operanzi cu un operator pentru a obţine o valoare nouă:

```
4 + 5
```

4 şi 5 sunt operanzii. Asupra lor operează operatorul de adunare (+) pentru a produce valoarea 9. Operatorii sunt, aproape întotdeauna, amplasaţi între cei doi operanzi, deşi veţi vedea, mai târziu în acest capitol, că există totuşi câteva excepţii.

Combinaţia operanzilor cu operatorul în vederea obţinerii unui rezultat se numeşte expresie. Deşi majoritatea operatorilor se află la baza expresiilor, o expresie nu trebuie să conţină neapărat un operator. De fapt, în PHP, o expresie se defineşte ca fiind orice poate fi folosit pe post de valoare. Astfel, expresiile includ constantele întregi, cum ar fi 654, variabilele, cum ar fi `$user` şi apelurile către funcţii, cum ar fi `gettype()`. `(4 + 5)`, de exemplu, este o expresie care constă din alte două expresii şi un operator. Atunci când o expresie produce o valoare, se spune, de obicei, că se rezolvă la respectiva valoare. Aşadar, atunci când sunt luate în considerare toate sub-expresiile, expresia poate fi tratată ca şi când ar fi un cod pentru respectiva valoare.

Apropo

> O expresie este orice combinaţie de funcţii, valori şi operatori care se rezolvă la o valoare. O regulă uşor de ţinut minte că dacă ceva poate fi folosit ca o valoare atunci acel ceva este o expresie.

Acum că am pus la punct problemele de principiu, a venit vremea să dăm o raită printre operatorii folosiţi în mod uzual în PHP.

Operatorul de atribuire

Aţi văzut operatorul de atribuire de fiecare dată când am iniţializat o variabilă; el constă dintr-un singur caracter =. Operatorul de atribuire preia valoarea operandului din partea dreaptă şi o atribuie operandului din partea stângă:

```
$name = "matt";
```

Variabila `$name` conţine acum şirul de caractere `"matt"`. În mod interesant, această construcţie este o expresie. La o primă vedere, s-ar părea că operatorul de atribuirea nu face altceva decât să schimbe valoarea variabilei `$name` fără a produce o valoare, dar, de fapt, instrucţiunile care folosesc operatorul de atribuire se rezolvă întotdeauna la o copie a valorii operatorului din partea dreaptă. Astfel:

```
echo $name = "matt";
```

va afişa pe ecran şirul de caractere `"matt"` în browser, pe lângă faptul că va atribui variabilei `$name` valoarea `"matt"`.

Operatori aritmetici

Operatorii aritmetici sunt exact ce vă aşteptaţi să fie — ei efectuează operaţii aritmetice. Tabelul 4.3 enumeră aceşti operatori. Operatorul de adunare adună operandul din stânga cu cel din dreapta. Operatorul de scădere scade operandul din dreapta din operandul din stânga. Operatorul de diviziune împarte operandul din stânga la operandul din dreapta. Operatorul de multiplicare înmulţeşte operandul din stânga cu cel din dreapta. Operatorul modulus returnează restul împărţirii operandului din stânga la operandul din dreapta.

TABELUL 4.3. Operatorii aritmetici.

Operator	Nume	Exemplu	Exemplu de rezultat
+	Adunare	1Ø+3	13
-	Scădere	1Ø-3	7
/	Diviziune	1Ø/3	3.3333333333333
*	Multiplicare	1Ø*3	3Ø
%	Modul	1Ø%3	1

Operatorul de concatenare

Operatorul de concatenare este reprezentat de un punct (.). Tratând ambii operanzi ca şiruri de caractere, acest operator ataşează operandul din partea dreaptă celui din partea stângă. Aşadar,

```
"hello"." world"
```

returnează

```
"hello world"
```

Observaţi că spaţiul dintre cele două cuvinte apare din cauză că există un spaţiu la începutul celui de-al doilea operand. Operatorul de concatenare efectiv lipeşte cele două şiruri de caractere, fără a adăuga nimic din proprie iniţiativă. Deci, dacă aţi fi încercat să concatenaţi două şiruri care nu ar fi conţinut spaţii la început sau la sfârşit, cum ar fi

```
"hello"."world"
```

aţi fi obţinut drept rezultat:

```
"helloworld"
```

Indiferent de tipurile de date ale operanzilor, aceştia sunt trataţi ca şi când ar fi şiruri de caractere şi, deci, rezultatul este întotdeauna un şir de caractere. Vom întâlni concatenarea destul de frecvent în această carte atunci când va trebui să combinăm rezultatele unei expresii de un fel sau altul cu un şir de caractere, ca în exemplul:

```
$centimeters = 212;
echo "the width is ".($centimeters/100)." meters";
```

Operatorii combinaţi de atribuire

Deşi există un singur operator de atribuire, PHP ne pune la dispoziţie o serie de operatori combinaţi care transformă operandul din partea stângă şi returnează un rezultat. De regulă, operatorii îşi folosesc operanzii fără să le schimbe valoarea, dar operatorii de atribuire încalcă această regulă. Un operator combinat de atribuire constă dintr-un operator standard urmat de semnul egal. Operatorii de atribuire combinaţi vă scapă de efortul de a tasta operandul din partea stângă de două ori. De exemplu:

```
$x = 4;
$x = $x + 4; //$x are acum valoarea 8
```

ar putea să fie scrisă şi sub forma

```
$x = 4;
$x += 4; //$x are acum valoarea 8
```

Există câte un operator de atribuire pentru fiecare dintre operatorii aritmetici şi unul pentru operatorul de concatenare. Tabelul 4.4 enumeră câţiva dintre cei mai utilizaţi operatori combinaţi.

TABELUL 4.4. Câţiva operatori combinaţi de atribuire.

Operator	Exemplu	Echivalent cu
+=	$x += 5	$x = $x + 5
-=	$x -= 5	$x = $x - 5
/=Ø	$x /= 5	$x = $x / 5
*=	$x *= 5	$x = $x * 5
%=	$x %= 5	$x = $x % 5

Fiecare dintre exemplele din tabelul 4.4 transformă valoarea lui $x folosind valoarea operandului din partea dreaptă.

Incrementarea şi decrementarea automată a unei variabile de tip `integer`

Atunci când scrieţi cod PHP, veţi avea de multe ori nevoie să incrementaţi sau să decrementaţi o variabilă de tip integer. De obicei veţi avea nevoie să faceţi acest lucru atunci când contorizaţi iteraţiile unei bucle. Aţi învăţat deja două modalităţi de a face acest lucru. Putem incrementa numărul întreg conţinut de variabila $x folosind operatorul de adunare:

```
$x = $x + 1; // $x este incrementat
```

sau un operator combinat de atribuire:

```
$x += 1; // $x este incrementat
```

În ambele cazuri, numărul întreg rezultat este atribuit variabilei $x. Deoarece expresiile de acest fel sunt atât de des întâlnite, PHP ne pune la dispoziţie câţiva operatori speciali care permit adăugarea sau scăderea constantei întregi 1 dintr-o variabilă de tip `integer`, atribuind rezultatul variabilei însăşi. Aceşti operatori sunt cunoscuţi sub numele post-decrement şi post-increment. Operatorul post-decrement constă din două simboluri plus anexate numelui variabilei.

```
$x++ //$x este incrementat
```

Această expresie incrementează variabila $x cu o unitate. Folosind două simboluri minus în acelaşi fel, putem decrementa valoarea variabilei:

```
$x-- // $x este decrementat
```

Dacă folosiţi operatorii post-increment şi post-decrement în conjuncţie cu un operator condiţional, operandul va fi modificat abia după ce testul este completat:

```
$x = 3;
$y = $x++ + 3;
```

În acest caz, $y devine mai întâi 6 (3 + 3) şi după aceea este incrementat $x.

În unele circumstanţe, aţi putea dori să incrementaţi sau să decrementaţi variabila dintr-o expresie condiţională înainte ca testul să fie efectuat. PHP vă pune la dispoziţie, în acest scop, operatorii pre-decrement şi pre-increment. Aceşti operatori se comportă exact ca post-increment şi post-decrement, dar simbolurile plus sau minus se scriu înaintea variabilei:

```
++$x; // $x este decrementat
--$x; // $x este decrementat
```

Dacă aceşti operatori sunt folosiţi în cadrul unei expresii condiţionale, incrementarea are loc înainte de efectuarea testului:

```
$x = 3;
++$x < 4; // fals
```

În fragmentul precedent, `$x` este incrementat înainte să fie testat comparativ cu 4. Expresia condiţională returnează false deoarece 4 nu este mai mic decât 4.

Operatori de comparare

Operatorii de comparare efectuează teste asupra operanzilor lor. Ei returnează valoarea de tip boolean `true` dacă testul este validat, şi `false` în caz contrar. Acest tip de expresii sunt utile în cadrul structurilor de control, cum ar fi instrucţiunile `if` şi `while`. Vom discuta despre aceste instrucţiuni în capitolul 5, care poartă numele *Funcţii de controlare a execuţiei codului în PHP.*

Pentru a verifica dacă valoarea conţinută de variabila `$x` este mai mică decât 5, de exemplu, puteţi folosi operatorul „mai mic decât":

```
$x < 5
```

Dacă `$x` conţine valoarea 3, această expresie va avea valoarea true. Dacă `$x` ar conţine valoarea 7 expresia s-ar rezolva la `false`.

Tabelul 4.5 enumeră operatorii de comparare.

TABELUL 4.5. Operatorii de comparare.

Operator	Nume	Returnează `true` dacă...	Exemplu ($x este 4)	Rezultat
`==`	Echivalență	*Stânga este echivalentă cu dreapta.*	`$x == 5`	`false`
`!=`	Non-echivalență	*Stânga nu este echivalentă cu dreapta.*	`$x != 5`	`true`
`===`	Identitate	*Stânga este echivalentă cu dreapta şi cele două sunt de acelaşi tip.*	`$x === 4`	`true`
	Non-echivalență	*Stânga este echivalentă cu dreapta, dar cele două nu sunt de acelaşi tip.*	`$x === "4"`	`false`
`>`	Mai mare decât	*Stânga este mai mare decât dreapta.*	`$x > 4`	`false`
`>=`	Mai mare sau egal cu	*Stânga este mai mare sau egală cu dreapta.*	`$x >= 4`	`true`
`<`	Mai mic decât	*Stânga este mai mică decât dreapta.*	`$x < 4`	`false`
`<=`	Mai mic sau egal cu	*Stânga este mai mică sau egală cu dreapta.*	`$x <= 4`	`true`

Aceşti operatori sunt, de obicei, folosiţi pentru valori de tip `integer` sau `double`, deşi operatorul de echivalenţă poate fi folosit la compararea şirurilor de caractere. Asiguraţi-vă că stăpâniţi diferenţa dintre operatorul == şi operatorul =. Operatorul == testează echivalenţa, pe când operatorul = atribuie valori.

Crearea unor expresii condiţionale mai complexe cu ajutorul operatorilor logici

Operatorii logici testează combinaţii de valori booleene. De exemplu, operatorul `or` (sau), care este reprezentat prin două caractere bară (||) sau pur şi simplu prin cuvântul `or`, returnează `true` dacă oricare dintre operanzi are valoarea `true`:

```
true || false
```

Această expresie returnează valoarea `true`.

Operatorul `and` (şi), care este reprezentat prin două caractere & (&&) sau, pur şi simplu, prin cuvântul `and`, returnează valoarea `true` atunci când ambii operanzi sunt adevăraţi:

```
true && false
```

Această expresie returnează valoarea `false`. Este puţin probabil că veţi folosi operatori logici pentru a testa constante booleene ci, mai degrabă, veţi testa expresii care se rezolvă la valori booleene. De exemplu,

```
( $x > 2 ) && ( $x < 15 )
```

returnează valoarea `true` dacă `$x` conţine o valoarea care este mai mare decât 2 şi mai mică decât 15. Am inclus între paranteze expresiile astfel încât codul să fie mai lizibil.

TABELUL 4.6. Operatori logici.

Operator	Nume	Returnează `true` dacă...	Exemplu ($x este 4)	Rezultat
`||`	Sau	*Dreapta sau stânga este adevărată.*	`true || false`	`true`
`or`	Sau	*Dreapta sau stânga este adevărată.*	`true or false`	`true`
`xor`	Sau exclusiv	*Dreapta sau stânga este adevărată, dar nu ambele în acelaşi timp.*	`true xor true`	`false`
`&&`	Şi	*Dreapta şi stânga sunt adevărate.*	`true && false`	`false`
`and`	Şi	*Dreapta şi stânga sunt adevărate.*	`true and false`	`false`
`!`	Non	*Singurul său operand este fals.*	`!true`	`false`

De ce există câte două versiuni atât pentru operatorul `or` cât şi pentru operatorul `and`? Răspunsul ţine de ordinea operaţiilor, pe care o vom discuta în continuare.

Ordinea operațiilor

Atunci când folosiţi un operator, motorul PHP citeşte expresia dumneavoastră, de obicei, de la stânga la dreapta. În cazul expresiilor complexe, care conţin mai mulţi operatori, totuşi, apele devin mai tulburi. Mai întâi, să luăm în discuţie un caz simplu:

```
4 + 5
```

Nu este loc de confuzie aici. PHP adună 4 cu 5. Dar ce părere aveţi de fragmentul următor?

```
4 + 5 * 2
```

Aici apare o problemă. Expresia aceasta înseamnă suma lui 4 cu 5, înmulţită cu 2 şi având drept rezultat 18? Sau înseamnă 4 plus rezultatul înmulţirii lui 5 cu 2, adică 14? În mod normal, citind expresia de la stânga la dreapta, prima versiune ar fi corectă. Dar PHP acordă priorităţi diferite diverşilor operatori, efectuând operaţiile într-o anumită ordine. Din cauză că operatorul de multiplicare are prioritate mai mare decât operatorul de adunare, cea de-a doua soluţie este cea corectă.

Nu este totul pierdut, însă. Puteţi folosi parantezele pentru a forţa motorul de PHP să execute suma înaintea înmulţirii:

```
(4 + 5) * 2
```

Oricare ar fi ordinea operaţiilor dintr-o expresie complexă, este o idee bună să folosiţi parantezele în scopul creşterii lizibilităţii codului şi al evitării greşelilor greu de descoperit. Mai jos se găseşte o listă cu operatorii discutaţi în acest capitol, în ordinea priorităţii lor (cei cu prioritate mai mare apar mai sus).

```
++, --, (cast)
/, *, %
+, -
<, <=, =>, >
==, ===, !=
&&
||
=, +=, -=, /=, *=, %=, .=
and
xor
or
```

După cum puteţi vedea, operatorul `or` are prioritate mai scăzută decât `||`, iar `and` are o prioritate mai scăzută decât `&&`, astfel încât să puteţi folosi operatorii cu prioritatea corespunzătoare pentru a schimba ordinea în care este citită o expresie complexă. Aceasta nu este o idee prea bună. Următoarele două expresii sunt echivalente, dar cea de-a doua este mult mai uşor de citit:

```
$x and $y || $z
$x && ($y || $z)
```

Mergând un pas mai departe, următoarea formă este şi mai lizibilă:

```
$x and ($y or $z)
```

Toate cele trei exemple de mai sus sunt echivalente.

Ordinea operaţiilor este singurul motiv pentru care atât `&&` cât şi `and` sunt prezenţi în PHP. Acelaşi lucru este valabil şi pentru `||` şi `or`. În aproape toate situaţiile, deşi nu în toate, folosirea parantezelor va contribui la claritatea codului şi la reducerea numărului de greşeli mai mult decât utilizarea diferenţei de prioritate a operatorilor. În această carte, vom folosi mai frecvent operatorii `||` şi `&&`.

Constante

Variabilele oferă o modalitate flexibilă de stocare a datelor. Puteţi să schimbaţi valorile şi tipul de date pe care le stochează în orice moment. Cu toate acestea, dacă vreţi să lucraţi cu o valoarea care doriţi să rămână neschimbată pe toată durata execuţiei scriptului, puteţi defini şi folosi o constantă. Trebuie să folosiţi funcţia PHP predefinită `define()` pentru a crea o constantă, a cărei valoare nu mai poate fi schimbată ulterior. Pentru a folosi funcţia `define()`, trebuie să amplasaţi numele constantei şi valoarea pe care doriţi să i-o atribuiţi între paranteze, separate prin virgulă.

```
define("NUME_CONSTANTA", 42);
```

Valoarea pe care doriţi să o atribuiţi poate fi un număr, un şir de caractere sau o valoare de tip `boolean`. Prin convenţie, numele constantelor trebuie să fie scrise cu majuscule. Constantele sunt accesibile numai prin intermediul numelui; nu este necesar simbolul dolar. Listingul 4.4 arată cum se defineşte şi accesează o constantă.

LISTINGUL 4.4. Definirea şi accesarea unei constante.

```
1: <html>
2: <head>
3: <title>Listing 4.4 Defining and accessing a constant</title>
4: </head>
5: <body>
```

LISTINGUL 4.4. (*continuare*)

```
 6: <?php
 7: define("USER", "Gerald");
 8: echo "Welcome ".USER;
 9: ?>
10: </body>
11: </html>
```

Apropo

Atunci când utilizați constante, țineți minte că ele pot fi folosite oriunde în scripturile dumneavoastră, inclusiv în funcțiile externe care sunt incluse.

Observaţi că, pe linia 8, am folosit operatorul de concatenare pentru a adăuga valoarea conţinută de constanta noastră şirului `"Welcome "`. Acest lucru este necesar deoarece motorul de PHP nu are cum să distingă între un şir simplu de caractere şi un nume de constantă dacă ambele sunt între ghilimele.

Puneţi aceste linii într-un fişier text numit `constant.php` şi amplasaţi fişierul în directorul rădăcină al serverului dumneavoastră Web. Atunci când accesaţi acest script cu ajutorul browserului, el produce următorul rezultat:

```
Welcome Gerald
```

Funcţia `define()` poate, de asemenea, să accepte şi un al treilea argument, de tip `boolean`, care spune dacă numele constantei trebuie să fie sau nu independent de capitalizare (folosirea literelor mari sau mici). Numele constantelor depind, implicit, de capitalizare. Cu toate acestea, dacă pasăm argumentul `true` funcţiei `define()`, putem schimba acest comportament, aşa că, dacă am iniţializa o constantă cu numele `USER` astfel:

```
define("USER", "Gerald", true);
```

am putea accesa valoarea constantei fără să conteze dacă folosim litere mari sau mici:

```
echo User;
echo usEr;
echo USER;
```

Aceste expresii sunt, toate, echivalente şi au drept rezultat afişarea şirului Gerald. Această caracteristică ar putea să îi ajute pe alţi programatori care ar lucra la scriptul dumneavoastră, deoarece aceştia nu ar trebui să ştie cum trebuie scrise numele constantelor definite de dumneavoastră pentru a le putea accesa. Pe de altă parte, dat fiind faptul că alte constante au nume care depind de capitalizare, această caracteristică ar putea să creeze, în loc să elimine, confuzie deoarece programatorii vor uita care constante trebuie tratate într-un anumit fel şi care într-altul. În afara cazului în care aveţi motive solide să nu o faceţi, trebuie întotdeauna să lăsaţi numele de constante să fie dependente de capitalizare şi să folosiţi mereu doar litere mari atunci când le definiţi, deoarece aceasta este convenţia cea mai uşor de reţinut.

Constante predefinite

PHP ne pune la dispoziţie automat o serie de constante predefinite. De exemplu, constanta `__FILE__` returnează numele fişierului pe care motorul de PHP îl citeşte în momentul respectiv. Constanta `__LINE__` returnează numărul liniei dintr-un fişier. Aceste constante sunt utile pentru generarea mesajelor de eroare. Puteţi, de asemenea, să aflaţi ce versiune de PHP interpretează scriptul, folosind constanta `PHP_VERSION`. Acest lucru poate fi util în cazul în care aveţi nevoie să afişaţi versiunea în rezultatul unui script atunci când trimiteţi un raport asupra unei greşeli de programare.

Rezumat

În acest capitol, am discutat despre câteva caracteristici de bază ale limbajului PHP. Aţi învăţat despre variabile şi cum să le atribuiţi valori folosind operatorul de atribuire, dar aţi beneficiat şi de o prezentare a unor concepte precum domeniu de definire şi variabile superglobale predefinite. Vi s-au mai prezentat şi operatorii şi aţi aflat cum să îi combinaţi pe cei mai uzuali dintre aceştia pentru a forma expresii. La final, aţi învăţat să definiţi şi să accesaţi constante.

Acum că aţi deprins câteva dintre elementele fundamentale ale programării în PHP, în următorul capitol chiar vă veţi aşeza pe scaunul şoferului. Veţi învăţa cum să faceţi scripturi care să ia decizii şi să repete operaţiuni cu ajutorul variabilelor, expresiilor şi al operatorilor.

Întrebări şi răspunsuri

Î. De ce este util să ştiu tipul datelor pe care le stochează o variabilă?

R. De multe ori, tipul datelor dintr-o variabilă limitează operaţiile pe care le puteţi face cu respectiva variabilă. S-ar putea să aveţi nevoie să vă asiguraţi că o variabilă conţine un număr de tip `integer` sau `double` înainte de a o folosi într-un calcul matematic, de exemplu.

Î. Ar trebui să respect vreo convenţie atunci când îmi denumesc variabilele?

R. Scopul dumneavoastră ar trebui să fie întotdeauna claritatea şi inteligibilitatea scriptului. O variabilă cu un nume de genul `$ab12345` nu vă spune nimic despre rolul ei în cadrul scriptului şi creşte riscul greşelilor de tastare. Păstraţi numele variabilelor cât mai scurte şi descriptive.

O variabilă numită `$f` nu are prea multe şanse să însemne ceva atunci când vă întoarceţi asupra codului după o lună. O variabilă `$nume_fisier`, pe de altă parte, are un înţeles clar.

Î. Ar trebui să învăţ pe de rost tabelul cu ordinea priorităţii operatorilor?

R. Nu există nici un motiv pentru care nu ar trebui să o faceţi, dar un astfel de efort ar merita o cauză mai nobilă. Folosind paranteze în expresiile dumneavoastră, veţi scrie un cod clar şi uşor de citit, definindu-vă singur ordinea operaţiilor.

Atelier

Atelierul este conceput să anticipeze posibile întrebări, să vă ajute să recapitulaţi cele învăţate şi să începeţi să vă puneţi cunoştinţele în practică.

Test

1. Care dintre următoarele nume de variabile nu este valid?

```
$a_value_submitted_by_a_user
$666666xyz
$xyz666666
$_____counter_____
$the first
$file-name
```

2. Care este rezultatul următorului fragment de cod?

```
$num = 33;
(boolean) $num;
echo $num;
```

3. Care este rezultatul următoarei instrucţiuni?

```
echo gettype("4");
```

4. Care ar fi rezultatul următorului fragment de cod?

```
$test_val = 5.5466;
settype($test_val, "integer");
echo $test_val;
```

5. Care dintre următoarele instrucţiuni nu conţine o expresie?

```
4;
gettype(44);
5/12;
```

6. Care dintre instrucţiunile de la punctul 5 conţine un operator?

7. Ce valoare va returna următoarea expresie?

```
5 < 2
```

Ce tip de date va avea valoarea returnată?

Soluţii

1. Numele de variabilă `$66666xyz` nu este valid deoarece nu începe cu o literă sau cu caracterul de subliniere. Numele variabilei `$the first` nu este valid deoarece conţine un spaţiu. `$file-name` nu este nici el un nume valid deoarece conţine un caracter non-alfanumeric.

2. Fragmentul de cod va afişa numărul întreg 33. Castingul la `boolean` produce o copie convertită a valorii variabilei `$num`, dar nu modifică valoarea care este efectiv stocată de această variabilă.

3. Instrucţiunea va afişa şirul de caractere `"string"`.

4. Codul va afişa valoarea 5. Când un număr de tip double este convertit la tipul integer, orice informaţii aflate dincolo de punctul zecimalelor se pierd.

5. Toate sunt expresii deoarece toate se rezolvă la valori.

6. Instrucţiunea 5/12; conţine un operator de diviziune.

7. Expresia se rezolvă la valoarea `false`, care este de tip `boolean`.

Activități

1. Creați un script care să conțină cel puțin cinci variabile diferite. Populați-le cu valori de diferite tipuri și folosiți funcția `gettype()` pentru a afișa tipul de date din fiecare variabilă în browser.
2. Atribuiți valori la două variabile. Folosiți operatorii de comparare pentru a verifica dacă prima valoare este:
   - ► aceeași ca cea de-a doua;
   - ► mai mică decât cea de-a doua;
   - ► mai mare decât cea de-a doua;
   - ► mai mică sau egală cu cea de-a doua.

   Afișați rezultatul fiecărui test în browser.

   Schimbați valorile atribuite variabilelor și rulați scriptul încă o dată.

CAPITOLUL 5

Funcții de controlare a execuției codului în PHP

Scripturile create în capitolul precedent se execută într-o singură direcție. Aceleași instrucțiuni sunt executate în aceeași ordine ori de câte ori este rulat scriptul. Din acest motiv, nu avem la dispoziție prea multă flexibilitate.

Acum veți învăța câteva structuri care le permit scripturilor dumneavoastră să se adapteze la situații diverse.

În acest capitol, veți învăța cum să:

- folosiți instrucțiunea if pentru a executa cod dacă o expresie condițională se rezolvă la valoarea `true`;
- executați blocuri de cod alternative atunci când expresia unei instrucțiuni `if` este evaluată la valoarea `false`;
- repetați execuția unui bloc de cod folosind instrucțiunea `while`;
- folosiți instrucțiuni `for` pentru a crea structuri repetitive mai elegante;
- părăsiți un bloc de cod repetitiv;
- imbricați o buclă în cadrul alteia;
- folosiți marcajele de deschidere și închidere a codului PHP în interiorul structurilor de control.

Alternarea codului de executat

Cele mai multe scripturi vor evalua condiții și își vor schimba comportamentul în mod corespunzător. Aceste decizii stau la baza paginilor dinamice ce pot fi create cu ajutorul scripturilor PHP; este vorba despre pagini care își pot schimba conținutul în funcție de situație. Similar altor limbaje de programare, PHP vă permite să faceți astfel de lucruri cu ajutorul instrucțiunii `if`.

Instrucțiunea if

O instrucțiune if reprezintă o modalitate de a controla execuția unei instrucțiuni care urmează (adică a unei singure instrucțiuni sau a unui bloc de cod pus între acolade). Instrucțiunea if evaluează o expresie pusă între paranteze. Dacă această expresie are valoarea true, instrucțiunea ce urmează este executată. Altminteri, instrucțiunea este omisă cu totul. Acest lucru le permite scripturilor să ia decizii pe baza unui număr de factori:

```
if(expresie){
        // cod de executat in cazul in care expresia este evaluata la
        // valoarea true
}
```

Listingul 5.1 execută un bloc de cod doar dacă o variabilă conține șirul de caractere "happy" (vesel).

LISTINGUL 5.1. O instrucțiune if.

```
 1: <html>
 2: <head>
 3: <title>Listing 5.1</title>
 4: </head>
 5: <body>
 6: <?php
 7: $mood = "happy";
 8: if ($mood == "happy") {
 9:     echo "Hooray, I'm in a good mood";
10: }
11: ?>
12: </body>
13: </html>
```

Folosim operatorul de comparare == pentru a compara variabila $mood (stare de spirit) cu șirul de caractere "happy". Dacă cele două sunt egale, expresia este evaluată la valoarea true, iar blocul de cod de dedesubtul instrucțiunii if este executat.

Puneți aceste linii într-un fișier text denumit testif.php și amplasați acest fișier în directorul rădăcină al serverului dumneavoastră Web. Atunci când accesați acest script prin intermediul browserului, el produce următorul rezultat:

```
Hooray, I'm in a good mood
```

Dacă schimbați valoarea variabilei $mood în "sad" (trist), sau în orice alt șir de caractere în afară de "happy" și, apoi, executați scriptul din nou, expresia condițională a instrucțiunii if va fi evaluată la valoarea false, iar blocul de cod va fi omis. Scriptul rămâne tăcut, ceea ce ne aduce la folosirea clauzei else.

Folosirea clauzei `else` împreună cu instrucțiunea `if`

Atunci când lucrați cu instrucțiuni `if`, de multe ori aveți nevoie să definiți un bloc de cod alternativ care să fie executat dacă expresia condițională este evaluată la valoarea `false`. Puteți face acest lucru prin adăugarea unei clauze `else` urmată de încă un bloc de cod:

```
if(expresie){
        // cod de executat in cazul in care expresia este evaluata la
        // valoarea true
} else {
        // cod de executat in orice alt caz
}
```

Listingul 5.2 este o versiune modificată a listingului 5.2 astfel încât blocul de cod implicit este executat în cazul în care `$mood` nu este echivalent cu `"happy"`.

LISTINGUL 5.2. O instrucțiune `if` care folosește clauza `else`.

```
 1: <html>
 2: <head>
 3: <title>Listing 5.2</title>
 4: </head>
 5: <body>
 6: <?php
 7: $mood = "sad";
 8: if ($mood == "happy") {
 9:     echo "Hooray, I'm in a good mood";
10: } else {
11:     echo "Not happy but $mood";
12: }
13: ?>
14: </body>
15: </html>
```

Puneți aceste linii într-un fișier text denumit `testifelse.php` și amplasați acest fișier în directorul rădăcină al serverului dumneavoastră Web. Atunci când accesați acest script prin intermediul browserului, el produce următorul rezultat:

```
Not happy but sad
```

Observați că, pe linia 7, variabila `$mood` conține șirul de caractere `"sad"`, care, evident, nu este egal cu `"happy"` așa că expresia condițională a instrucțiunii `if` de pe linia 8 este evaluată la valoarea `false`. Din această cauză, primul bloc de cod (linia 9) este omis. Cu toate acestea, blocul de cod de dedesubtul clauzei `else` este executat, iar mesajul `Not happy but sad` (nu vesel ci trist) este afișat în browser. Șirul de caractere `"sad"` reprezintă valoarea atribuită variabilei `$mood`.

Folosirea clauzei `else` împreună cu instrucţiunea `if` le permite scripturilor să ia decizii sofisticate, dar opţiunile sunt limitate, momentan, la o ramificare dihotomică. PHP vă permite să evaluaţi mai multe expresii una după cealaltă, aşa cum veţi învăţa imediat.

Folosirea clauzei `else if` împreună cu instrucţiunea `if`

Puteţi să folosiţi o construcţie `if...else if...else` pentru a verifica mai multe expresii înainte de a oferi un bloc de cod implicit:

```
if(expresie){
        // cod de executat in cazul in care expresia este evaluata la
        // valoarea true
} else if(o alta expresie) {
        // cod de executat in cazul in care conditia precedenta nu a
        // fost indeplinita, iar expresia este evaluata la true
} else {
        // cod de executat in orice alt caz
}
```

Dacă prima expresie nu este evaluată la valoarea `true`, primul bloc de cod este ignorat. Clauza `else if` evaluează, apoi, o altă expresie. Din nou, dacă această expresie este evaluată la valoarea `true`, cel de-al doilea bloc de cod este executat. Altminteri, blocul de cod asociat clauzei `else` este executat. Puteţi include oricât de multe clauze `else if` doriţi şi, în cazul în care nu aveţi nevoie de o acţiune implicită, puteţi să nu scrieţi nici o clauză `else`.

Apropo

> Clauza `else if` poate fi scrisă şi sub forma unui singur cuvânt (`elseif`). Puteţi folosi oricare dintre aceste două sintaxe, după gust.

Listingul 5.3 adaugă o clauză `else if` la exemplul precedent.

LISTINGUL 5.3. O instrucţiune `if` care foloseşte atât `else`, cât şi `else if`.

```
 1: <html>
 2: <head>
 3: <title>Listing 5.3</title>
 4: </head>
 5: <body>
 6: <?php
 7: $mood = "sad";
 8: if ($mood == "happy") {
 9:     echo "Hooray, I'm in a good mood";
10: } else if ($mood == "sad") {
11:     echo "Awww. Don't be down!";
12: } else {
13:     echo "Neither happy nor sad but $mood";
14: }
15: ?>
16: </body>
17: </html>
```

Şi de data aceasta, `$mood` conţine un şir de caractere, `"sad"`, care îi este atribuit pe linia 7. Acest şir nu este egal cu `"happy"`, aşa că primul bloc de cod, de pe linia 9, este ignorat. Clauza `else if` de pe linia 10 verifică echivalenţa dintre conţinutul variabilei `$mood` şi valoarea `"sad"`, care se evaluează la valoarea `true`. Acest bloc de cod este, deci, executat. Pe liniile 12, 13 şi 14 punem la punct comportamentul implicit al scriptului care este invocat atunci când nici un a dintre condiţii nu este îndeplinită. În acest caz, tot ce facem este să afişăm un mesaj ce include valoarea efectivă a variabilei `$mood`.

Puneţi aceste linii într-un fişier text denumit `testifelseif.php` şi amplasaţi acest fişier în directorul rădăcină al serverului dumneavoastră Web. Atunci când accesaţi acest script prin intermediul browserului, el produce următorul rezultat:

```
Awww. Don't be down!
```

Schimbaţi valoarea lui `$mood` în `"unkown"` (necunoscut) şi rulaţi scriptul. El va produce următorul rezultat:

```
Neither happy nor sad but unknown
```

Instrucțiunea `switch`

Instrucţiunea `switch` este o metodă alternativă de a schimba cursul execuţiei în funcţie de valoarea unei expresii. Folosind instrucţiunea `if` în conjuncţie cu `else if`, puteţi evalua mai multe expresii. Pe de altă parte, instrucţiunea `switch` evaluează o aceeaşi expresie, executând blocuri de cod diferite în funcţie de rezultatul acelei expresii, atâta vreme cât expresia poate fi evaluată sub forma unui tip de date simplu (un număr, un şir de caractere sau o valoare booleană). Rezultatul unei expresii evaluate în cadrul unei instrucţiuni `if` este citit ca fiind fie `true`, fie `false`, pe când expresia folosită de instrucţiunea `switch` chiar este evaluată la un rezultat ce este comparat, apoi, cu o serie de valori:

```
switch(expresie){
        case resultat1:
                // cod ce se executa daca rezultatul expresiei este rezultat1
                break;
        ...
        default:
                // cod ce se executa daca nu a fost intalnita
                // nici o instructiune break pana aici
}
```

Expresia folosită în cadrul unei instrucţiuni `switch` este, de cele mai multe ori, o variabilă. În interiorul blocului de cod al instrucţiunii `switch`, se găseşte un număr de instrucţiuni `case`. Fiecare dintre acestea compară rezultatul expresiei din instrucţiunea `switch` cu o valoare. Dacă expresia este echivalentă cu valoarea din instrucţiunea `case`, este executat codul de după instrucţiunea `case`.

Instrucţiunea `break` întrerupe execuţia întregii instrucţiuni `switch`. Dacă instrucţiunea `break` este omisă, va fi evaluată şi următoarea instrucţiune `case`. Dacă se ajunge la instrucţiunea `default`, este executat codul de după aceasta.

Atenţie!

> Este important să includeţi o instrucţiune `break` la sfârşitul fiecărui bloc de cod care va fi executat ca parte dintr-o instrucţiune `case`. În lipsa instrucţiunii `break`, programul va continua să se execute cu următoarele instrucţiuni `case` şi, în final, cu instrucţiunea `default`. În cele mai multe cazuri, acest lucru va produce un comportament neaşteptat şi, foarte probabil, incorect din partea scriptului.

Listingul 5.4 recreează funcţionalitatea exemplului de folosire a instrucţiunii `if`, dar folosind instrucţiunea `switch`.

LISTINGUL 5.4. O instrucţiune `switch`.

```
 1: <html>
 2: <head>
 3: <title>Listing 5.4</title>
 4: </head>
 5: <body>
 6: <?php
 7: $mood = "sad";
 8: switch ($mood) {
 9:     case "happy":
10:         echo "Hooray, I'm in a good mood";
11:         break;
12:     case "sad":
13:         echo "Awww. Don't be down!";
14:         break;
15:     default:
16:         print "Neither happy nor sad but $mood";
17:         break;
18: }
19: ?>
20: </body>
21: </html>
```

Şi de această dată, pe linia 7, variabila `$mood` este iniţializată la valoarea `"sad"`. Instrucţiunea `switch` de pe linia 8 foloseşte această variabilă ca expresie a sa. Prima instrucţiune `case`, de pe linia 9, verifică echivalenţa dintre `"happy"` şi variabila `$mood`. Cele două nu se potrivesc, aşa că execuţia trece mai departe, la cea de-a doua instrucţiune `case`, de pe linia 12. Şirul `"sad"` este echivalent cu valoarea variabilei `$mood`, aşa că acest bloc de cod este executat. Instrucţiunea `break` de pe linia 14 opreşte procesul. Liniile de la 15 la 17 definesc acţiunea implicită care s-ar executa în cazul în care nici una dintre cele două condiţii ale instrucţiunilor `case` nu ar fi îndeplinită.

Puneţi aceste linii într-un fişier text denumit `testswitch.php` şi amplasaţi acest fişier în directorul rădăcină al serverului dumneavoastră Web. Atunci când accesaţi acest script prin intermediul browserului, se obţine următorul rezultat:

```
Awww. Don't be down!
```

Schimbaţi valoarea variabilei `$mood` în `"happy"` şi executaţi scriptul; se va obţine următorul rezultat:

```
Hooray, I'm in a good mood
```

Pentru a sublinia importanţa instrucţiunii `break`, încercaţi să executaţi scriptul fără cea de-a doua instrucţiune `break`. Rezultatul va fi:

```
Awww. Don't be down!Neither happy nor sad but sad
```

Acesta nu este în nici un caz rezultatul dorit, aşa că aveţi grijă să includeţi instrucţiuni `break` oriunde este nevoie de ele!

Folosirea operatorului ?

Operatorul `?`, numit şi operator ternar, este similar instrucţiunii `if`, numai că returnează o valoarea pe baza uneia dintre cele două expresii care urmează după el şi care sunt separate prin caracterul două puncte. Această construcţie are trei părţi, de unde şi numele ternar. Expresia folosită pentru generarea valorii returnate depinde de rezultatul expresiei condiţionale:

```
(expresie)?valoare_daca_expresia_este_true:valoare_daca_expresia_este_false
```

Dacă expresia condiţională este evaluată la `true`, este returnat rezultatul celei de-a doua expresii; în caz contrar, valoarea celei de-a treia expresii este returnată. Listingul 5.5 foloseşte operatorul ternar pentru a da o valoarea unei variabile în funcţie de valoarea variabilei `$mood`.

LISTINGUL 5.5. Folosirea operatorului ?.

```
 1: <html>
 2: <head>
 3: <title>Listing 5.5</title>
 4: </head>
 5: <body>
 6: <?php
 7: $mood = "sad";
 8: $text = ($mood == "happy") ? "I'm in a good mood" : "Not happy but $mood";
 9: echo "$text";
10: ?>
11: </body>
12: </html>
```

Pe linia 7, `$mood` capătă valoarea `"sad"`. Pe linia 8, este verificată echivalenţa variabilei `$mood` cu şirul `"happy"`. Din moment ce acest test returnează valoarea `false`, operatorul ternar va returna la rândul său rezultatul celei de-a treia expresii.

Puneţi aceste linii într-un fişier text denumit `testtern.php` şi amplasaţi acest fişier în directorul rădăcină al serverului dumneavoastră Web. Atunci când accesaţi acest script prin intermediul browserului, el produce următorul rezultat:

```
Not happy but sad
```

Operatorul ternar poate fi dificil de citit, dar este folositor când aveţi de-a face cu două alternative şi vreţi să scrieţi cod compact.

Structuri repetitive

Până acum, am discutat despre deciziile pe care le poate lua un script asupra codului pe care să îl execute. Scripturile mai pot decide şi de câte ori să execute un bloc de cod. Instrucţiunile iterative sunt concepute pentru a vă permite să efectuaţi sarcini repetitive. O buclă iterativă va continua să ruleze atâta timp cât o condiţie este îndeplinită, sau până când alegeţi să părăsiţi bucla în mod explicit.

Instrucţiunea `while`

Instrucţiunea `while` are o structură similară cu o instrucţiune `if` simplă:

```
while(expresie){
      // executa ceva
}
```

Totuşi, spre deosebire de instrucţiunea `if`, o instrucţiune `while` se va executa când expresia este evaluată la `true`, din nou şi din nou, de oricâte ori este nevoie. Fiecare execuţie a blocului de cod din cadrul buclei se numeşte iteraţie. În cadrul blocului de cod, de obicei, schimbaţi ceva care afectează expresia instrucţiunii `while`; dacă nu faceţi acest lucru, bucla se va executa la infinit. Listingul 5.6 creează o buclă `while` care calculează şi afişează multiplii de 2, până la 24.

LISTINGUL 5.6. O instrucţiune `while`.

```
 1: <html>
 2: <head>
 3: <title>Listing 5.6</title>
 4: </head>
 5: <body>
 6: <?php
 7: $counter = 1;
 8: while ($counter <= 12) {
 9:     echo "$counter times 2 is ".($counter * 2)."<br>";
10:     $counter++;
11: }
12: ?>
13: </body>
14: </html>
```

În acest exemplu, am inițializat variabila `$counter` pe linia 7. Instrucțiunea `while` de pe linia 8 verifică variabila `$counter`. Atâta vreme cât numărul conținut în variabila `$counter` este mai mic sau egal cu 12, bucla continuă să se execute. În cadrul blocului de cod al instrucțiunii `while`, valoarea conținută de variabila `$counter` este multiplicată cu doi, iar rezultatul este afișat în browser. Apoi, valoarea variabilei `$counter` este incrementată pe linia 10. Această etapă este extrem de importantă. Dacă ați uita să schimbați valoarea atribuită variabilei `$counter`, expresia `while` nu s-ar rezolva niciodată la valoarea false, iar bucla ar continua la nesfârșit.

Puneți aceste linii într-un fișier text denumit `testwhile.php` și amplasați acest fișier în directorul rădăcină al serverului dumneavoastră Web. Atunci când accesați acest script prin intermediul browserului, el produce următorul rezultat:

```
1 times 2 is 2
2 times 2 is 4
3 times 2 is 6
4 times 2 is 8
5 times 2 is 10
6 times 2 is 12
7 times 2 is 14
8 times 2 is 16
9 times 2 is 18
10 times 2 is 20
11 times 2 is 22
12 times 2 is 24
```

Instrucțiunea `do...while`

O instrucțiune `do...while` arată ca o instrucțiune `while` așezată cu capul în jos. Diferența esențială dintre cele două este că blocul de cod, în cazul instrucțiunii `do...while`, este executat înainte de testarea condiției și nu după aceea:

```
do {
        // cod de executat
} while(expresie);
```

Apropo

Expresia condițională a unei instrucțiuni `do...while` trebuie încheiată întotdeauna cu punct și virgulă.

Acest tip de instrucțiune este utilă atunci când vreți ca blocul de cod să se execute cel puțin o dată, chiar dacă expresia condițională a instrucțiunii `while` se rezolvă la `false`. Listingul 5.7 creează o instrucțiune `do...while`. Blocul de cod este executat cel puțin o dată.

LISTINGUL 5.7. Instrucțiunea `do...while`.

```
 1: <html>
 2: <head>
 3: <title>Listing 5.7</title>
 4: </head>
 5: <body>
 6: <?php
 7: $num = 1;
 8: do {
 9:     echo "The number is: $num<br>\n";
10:     $num++;
11: } while (($num > 200) && ($num < 400));
12: ?>
13: </body>
14: </html>
```

Instrucțiunea `do...while` verifică dacă variabila `$num` conține o valoarea care este mai mare decât `200` și mai mică decât `400`. Pe linia 7, am inițializat variabila `$num` la valoarea `1`, așa că expresia condițională va returna valoarea `false`. Cu toate acestea, blocul de cod este executat înainte ca expresia să fie evaluată, așa că instrucțiunea va afișa o singură linie în browser.

Puneți aceste linii într-un fișier text denumit `testdowhile.php` și amplasați acest fișier în directorul rădăcină al serverului dumneavoastră Web. Atunci când accesați acest script prin intermediul browserului, el produce următorul rezultat:

```
The number is: 1
```

Dacă schimbați valoarea variabilei `$num`, pe linia 7, în ceva de genul `300` și apoi executați scriptul, acesta va afișa:

```
The number is: 300
```

și va continua să afișeze linii similare, cu numere din ce în ce mai mari, până la:

```
The number is: 399
```

Instrucțiunea `for`

Orice se poate face cu ajutorul instrucțiunii `for` se poate face și cu o instrucțiune `while`, dar instrucțiunea `for` este, de cele mai multe ori, o cale mai elegantă și mai sigură de a obține același rezultat. Mai sus, în listingul 5.6, am inițializat o variabilă înainte de instrucțiunea `while`. Instrucțiunea `while` a testat, apoi, variabila în cadrul expresiei sale. Variabila a fost, ulterior, incrementată în cadrul blocului de cod. Instrucțiunea `for` vă permite să faceți toate aceste lucruri pe o singură linie. Din acest motiv, codul va fi mai compact și va fi mai puțin probabil să uitați să incrementați variabila contor, fapt ce ar duce la o buclă infinită:

```
for(expresie de initializare; expresie conditionala; expresie de
    modificare) {
        // cod de executat
}
```

Apropo

Buclele infinte, după cum le spune şi numele, sunt bucle care rulează la nesfârşit. Dacă bucla dumneavoastră este infinită, scriptul va rula o perioadă nedefinită de timp. Acest lucru împovărează excesiv serverul Web, iar pagina Web respectivă este inutilizabilă.

Expresiile din paranteza instrucţiunii `for` sunt separate prin caractere punct şi virgulă. De obicei, prima expresie iniţializează variabila contor, a doua expresie reprezintă testul condiţional al buclei, iar a treia expresie incrementează contorul. Listingul 5.8 arată o instrucţiune `for` care recreează exemplul din listingul 5.6, înmulţind 12 numere cu 2.

LISTINGUL 5.8. Folosirea funcției `for`.

```
 1: <html>
 2: <head>
 3: <title>Listing 5.8</title>
 4: </head>
 5: <body>
 6: <?php
 7: for ($counter=1; $counter<=12; $counter++) {
 8:     echo "$counter times 2 is ".($counter * 2)."<br>";
 9: }
10: ?>
11: </body>
12: </html>
```

Puneţi aceste linii într-un fişier text denumit `testfor.php` şi amplasaţi acest fişier în directorul rădăcină al serverului dumneavoastră Web. Atunci când accesaţi acest script prin intermediul browserului, el produce următorul rezultat:

```
1 times 2 is 2
2 times 2 is 4
3 times 2 is 6
4 times 2 is 8
5 times 2 is 10
6 times 2 is 12
7 times 2 is 14
8 times 2 is 16
9 times 2 is 18
10 times 2 is 20
11 times 2 is 22
12 times 2 is 24
```

Rezultatele listingului 5.6 şi 5.8 sunt identice, dar codul din listingul 5.8 este mai compact datorită folosirii instrucţiunii `for`. Deoarece variabila `$counter` este iniţializată şi incrementată la începutul buclei, logica acesteia este clară dintr-o

privire. Pe linia 7, prima expresie iniţializează variabila `$counter` la valoarea 1. Expresia condiţională verifică dacă variabila `$counter` conţine o valoare mai mică sau egală cu 12. Expresia finală incrementează variabila `$counter`.

Atunci când execuţia programului ajunge la bucla `for`, variabila `$counter` este iniţializată, iar expresia condiţională este evaluată. Dacă expresia este evaluată la valoarea `true`, blocul de cod este executat. Variabila `$counter` este, apoi, incrementată şi expresia condiţională este evaluată din nou. Acest proces continuă până când expresia condiţională este evaluată la valoarea `false`.

Părăsirea buclelor cu ajutorul instrucţiunii `break`

Atât instrucţiunile `while` cât şi cele `for` incorporează o expresie condiţională cu ajutorul căreia puteţi opri repetarea buclei. Pe lângă asta, instrucţiunea `break` vă permite să părăsiţi o buclă pe baza rezultatelor unor teste adiţionale. Acest lucru se poate dovedi o măsură de siguranţă împotriva erorilor. Listingul 5.9 creează o instrucţiune `for` simplă care împarte un număr mare la o variabilă care este incrementată, afişând rezultatele pe ecran.

LISTINGUL 5.9. O buclă `for` care împarte numărul 4000 la 10 numere crescătoare.

```
 1: <html>
 2: <head>
 3: <title>Listing 5.9</title>
 4: </head>
 5: <body>
 6: <?php
 7: for ($counter=1; $counter <= 10; $counter++) {
 8:     $temp = 4000/$counter;
 9:     echo "4000 divided by $counter is... $temp<br>";
10: }
11: ?>
12: </body>
13: </html>
```

Pe linia 7, exemplul iniţializează variabila `$counter` la valoarea 1. Expresia condiţională a instrucţiunii `for` verifică dacă variabila `$counter` este mai mică sau egală cu 10. În cadrul blocului de cod, 4000 este împărţit la `$counter`, iar rezultatul este afişat pe ecran.

Puneţi aceste linii într-un fişier text denumit `testfor2.php` şi amplasaţi acest fişier în directorul rădăcină al serverului dumneavoastră Web. Atunci când accesaţi acest script prin intermediul browserului, el produce următorul rezultat:

```
4000 divided by 1 is... 4000
4000 divided by 2 is... 2000
4000 divided by 3 is... 1333.33333333
```

```
4000 divided by 4 is... 1000
4000 divided by 5 is... 800
4000 divided by 6 is... 666.666666667
4000 divided by 7 is... 571.428571429
4000 divided by 8 is... 500
4000 divided by 9 is... 444.444444444
4000 divided by 10 is... 400
```

Acest script pare cât se poate de corect. Dar ce s-ar întâmpla dacă valoarea la care inițializați variabila `$counter` ar fi introdusă de utilizator? Valoarea respectivă ar putea să fie un număr negativ, sau chiar un șir de caractere. Haideți să analizăm prima situație. Dacă schimbăm valoarea inițială a variabilei `$counter` din 1 în –4, numărul 4000 va ajunge să fie împărțit la 0 atunci când blocul de cod este executat a cincea oară. De obicei, nu este o idee bună să împărțiți la zero în codul dumneavoastră, deoarece rezultatul este „nedefinit". Codul din listingul 5.10 se protejează de o astfel de eventualitate părăsind bucla iterativă dacă variabila `$counter` ajunge la valoarea 0.

LISTINGUL 5.10. Folosirea instrucțiunii `break`.

```
 1: <html>
 2: <head>
 3: <title>Listing 5.10</title>
 4: </head>
 5: <body>
 6: <?php
 7: $counter = -4;
 8: for (; $counter <= 10; $counter++) {
 9:     if ($counter == 0) {
10:         break;
11:     } else {
12:         $temp = 4000/$counter;
13:         echo  "4000 divided by $counter is... $temp<br>";
14:     }
15: }
16 ?>
17: </body>
18: </html>
```

Apropo

Împărțirea unui număr la zero nu provoacă o eroare irecuperabilă în PHP. Motorul de PHP doar generează un avertisment și continuă execuția.

Folosim o instrucțiune `if`, pe linia 9, pentru a verifica valoarea variabilei `$counter`. Dacă aceasta este egală cu zero, instrucțiunea `break` oprește imediat execuția blocului de cod, iar programul continuă să ruleze începând cu prima linie de după bucla `for` (linia 16).

Puneți aceste linii într-un fișier text denumit `testfor3.php` și amplasați acest fișier în directorul rădăcină al serverului dumneavoastră Web. Atunci când accesați acest script prin intermediul browserului, el produce următorul rezultat:

```
4000 divided by -4 is... -1000
4000 divided by -3 is... -1333.33333333
4000 divided by -2 is... -2000
4000 divided by -1 is... -4000
```

Observaţi că am iniţializat variabila `$counter` pe linia 7, în afara parantezei instrucţiunii `for`, pentru a simula situaţia în care valoarea acestei variabile este primită din afara scriptului.

Ştiaţi că...

Puteţi omite oricare dintre expresiile instrucţiunii `for`, dar trebuie să păstraţi întotdeauna caracterele punct şi virgulă la locul lor.

Oprirea unei singure iteraţii cu ajutorul instrucţiunii `continue`

Instrucţiunea `continue` opreşte execuţia iteraţiei curente, dar nu provoacă oprirea întregii bucle. Aşadar, următoarea iteraţie începe imediat. Folosirea instrucţiunii `break`, aşa cum am făcut în listingul 5.10, este o măsură drastică. Cu ajutorul instrucţiunii `continue`, aşa cum se vede în listingul 5.11, putem să evităm împărţirea la zero fără a părăsi bucla complet.

LISTINGUL 5.11. Folosirea instrucţiunii `continue`.

```
 1: <html>
 2: <head>
 3: <title>Listing 5.11</title>
 4: </head>
 5: <body>
 6: <?php
 7: $counter = -4;
 8: for (; $counter <= 10; $counter++) {
 9:     if ($counter == 0) {
10:         continue;
11:     }
12:     $temp = 4000/$counter;
13:     echo "4000 divided by $counter is... $temp<br>";
14: }
15: ?>
16: </body>
17: </html>
```

Pe linia 10, am înlocuit instrucţiunea `break` cu o instrucţiune `continue`. Dacă variabila `$counter` ajunge la valoarea zero, iteraţia este sărită şi următoarea iteraţie începe imediat.

Puneţi aceste linii într-un fişier text denumit `testcontinue.php` şi amplasaţi acest fişier în directorul rădăcină al serverului dumneavoastră Web. Atunci când accesaţi acest script prin intermediul browserului, se produce următorul rezultat:

```
4000 divided by -4 is... -1000
4000 divided by -3 is... -1333.33333333
4000 divided by -2 is... -2000
4000 divided by -1 is... -4000
4000 divided by 1 is... 4000
4000 divided by 2 is... 2000
4000 divided by 3 is... 1333.33333333
4000 divided by 4 is... 1000
4000 divided by 5 is... 800
4000 divided by 6 is... 666.666666667
4000 divided by 7 is... 571.428571429
4000 divided by 8 is... 500
4000 divided by 9 is... 444.444444444
```

Atenție!

Instrucțiunile `break` și `continue` pot diminua lizibilitatea scriptului. Din cauză că folosirea lor adaugă niveluri suplimentare de complexitate raționamentului din spatele instrucțiunilor iterative, ar trebui să le folosiți cu grijă.

Imbricarea structurilor repetitive

Instrucţiunile iterative pot conţine, în interior, alte instrucţiuni iterative. Astfel de combinaţii ale acestor instrucţiuni pot fi extrem de utile atunci când creaţi dinamic tabele HTML. În listingul 5.12 se folosesc două instrucţiuni `for` pentru a afişa o tablă a înmulţirii în browser.

LISTINGUL 5.12. Două bucle `for` imbricate.

```
 1: <html>
 2: <head>
 3: <title>Listing 5.12</title>
 4: </head>
 5: <body>
 6: <?php
 7: echo "<table border=\"1\"> \n";
 8: for ($y=1; $y<=12; $y++) {
 9:     echo "<tr> \n";
10:     for ($x=1; $x<=12; $x++) {
11:         echo "<td>";
12:         echo ($x * $y);
13:         echo "</td> \n";
14:     }
15:     echo "</tr> \n";
16: }
17: echo "</table>";
18: ?>
19: </body>
20: </html>
```

Înainte de a examina cele două bucle `for`, haideţi să ne uităm mai îndeaproape la linia 7 din listingul 5.12:

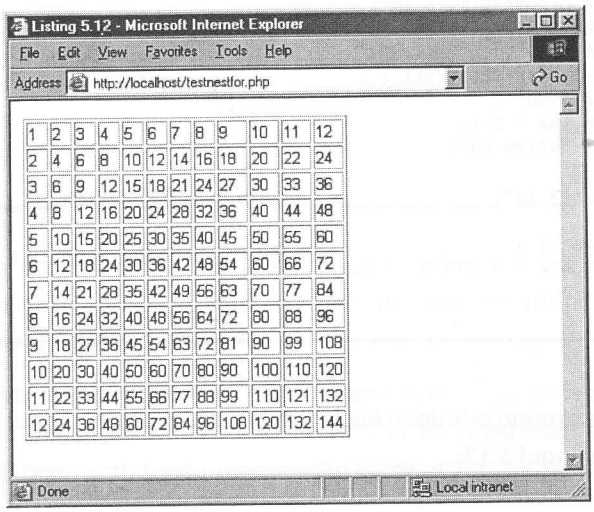
```
echo "<table border=\"1\"> \n";
```

Observaţi că am folosit caracterul de eludare (\) înainte de înainte de fiecare semn de citare din cadrul şirului de caractere. Acest lucru este necesar pentru ca motorul de PHP să ştie că vrem să folosim caracterul de citare în interiorul şirului şi că, deci, acesta nu trebuie interpretat ca început sau sfârşit al şirului de caractere. Dacă nu am face acest lucru, instrucţiunea nu ar fi înţeleasă de către motorul de PHP, deoarece ar citi-o ca pe un şir, urmat de un număr, urmat de încă un şir. Această situaţie ar provoca o eroare. În acest listing, mai folosim şi secvenţa de caractere `\n`, care reprezintă un caracter de linie nouă.

Instrucţiunea `for` exterioară (linia 8) iniţializează o variabilă denumită `$y`, atribuindu-i o valoare iniţială de `1`. Instrucţiunea defineşte şi o expresie care verifică dacă valoarea variabilei `$y` este mai mică sau egală cu `12` şi, cea de-a treia expresie, defineşte incrementarea variabilei `$y`. La fiecare iteraţie, blocul de cod afişează un marcaj HTML `<tr>` (rând de tabel) — pe linia 9 — şi defineşte o nouă instrucţiune `for` — pe linia 10. Această buclă internă iniţializează o variabilă denumită `$x` şi defineşte expresiile la fel ca cele ale buclei exterioare. La fiecare iteraţie, bucla internă afişează un marcaj <td> (celulă de tabel) în browser — pe linia 11 — dar şi rezultatul înmulţirii valorii variabilei `$x` cu cea a variabilei `$y` — pe linia 12. Pe linia 13, închidem tabelul celula tabelului HTML. După ce bucla internă se termină, ajungem iarăşi în blocul de cod al buclei exterioare, în cadrul căruia închidem rândul tabelului HTML pe linia 15. Acum totul este gata pentru ca procesul să înceapă din nou. Atunci când şi bucla exterioară se termină, rezultatul este o tablă a înmulţirii formatată frumos. Ne terminăm treaba închizând şi tabelul HMTL, pe linia 17.

Puneţi aceste linii într-un fişier text denumit `testnestfor.php` şi amplasaţi acest fişier în directorul rădăcină al serverului dumneavoastră Web. Atunci când accesaţi acest script prin intermediul browserului, se produce un rezultat asemănător cu cel din figura 5.1.

FIGURA 5.1.
Rezultatul listingului 5.12.



1	2	3	4	5	6	7	8	9	10	11	12
2	4	6	8	10	12	14	16	18	20	22	24
3	6	9	12	15	18	21	24	27	30	33	36
4	8	12	16	20	24	28	32	36	40	44	48
5	10	15	20	25	30	35	40	45	50	55	60
6	12	18	24	30	36	42	48	54	60	66	72
7	14	21	28	35	42	49	56	63	70	77	84
8	16	24	32	40	48	56	64	72	80	88	96
9	18	27	36	45	54	63	72	81	90	99	108
10	20	30	40	50	60	70	80	90	100	110	120
11	22	33	44	55	66	77	88	99	110	121	132
12	24	36	48	60	72	84	96	108	120	132	144

Blocurile de cod şi datele afişate în browser

În capitolul 3, intitulat *Instalarea şi configurarea pachetului PHP*, aţi învăţat că puteţi să treceţi din modul HTML în modul PHP şi înapoi, când şi de câte ori vreţi, folosind marcajele de deschidere şi închidere a codului PHP. În acest capitol, aţi descoperit că puteţi trimite date diferite către browser în funcţie de rezultatele procesului de luare a deciziilor pe care îl putem controla cu ajutorul instrucţiunilor `if` şi `switch`. În această secţiune, vom combina aceste două tehnici.

Imaginaţi-vă un script care afişează un tabel cu valori doar atunci când o variabilă booleană are valoarea `true`. Listingul 5.13 vă arată un tabel HTML simplificat construit în interiorul blocului de cod al unei instrucţiuni `if`.

LISTINGUL 5.13. Un bloc de cod conținând mai multe instrucțiuni `echo`.

```
 1: <html>
 2: <head>
 3: <title>Listing 5.13</title>
 4: </head>
 5: <body>
 6: <?php
 7: $display_prices = true;
 8: if ($display_prices) {
 9:     echo "<table border=\"1\">";
10:     echo "<tr><td colspan=\"3\">";
11:     echo "today's prices in dollars";
12:     echo "</td></tr>";
13:     echo "<tr><td>14</td><td>32</td><td>71</td></tr>";
14:     echo "</table>";
15: }
16: ?>
17: </body>
18: </html>
```

Dacă variabila `$display_prices` (afişează preţurile) capătă valoarea `true` pe linia 7, tabelul este afişat. În scopul claritătii, am fragmentat datele afişate în mai multe instrucţiuni `echo` şi am folosit, din nou, caractere de scăpare pentru a evita procesarea ghilimelelor.

Puneţi aceste linii într-un fişier text denumit `testmultiprint.php` şi amplasaţi acest fişier în directorul rădăcină al serverului dumneavoastră Web. Atunci când accesaţi acest script prin intermediul browserului, se produce rezultatul afişat în figura 5.2.

Nu este nimic greşit în această modalitate de scriere a codului, dar am putea să scăpăm de atâta tastat trecând înapoi în modul HTML în interiorul blocului de cod `if`. În listingul 5.14, facem exact acest lucru.

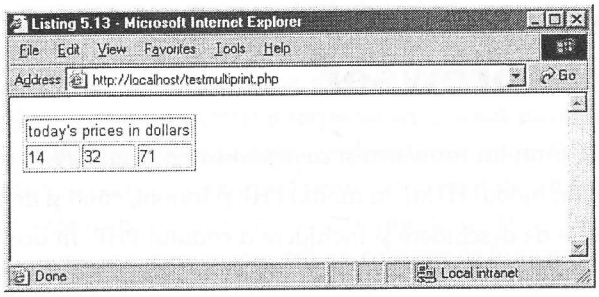


FIGURA 5.2. Rezultatul listingului 5.13.

LISTINGUL 5.14. Revenirea în modul HTML în interiorul unui bloc de cod.

```
 1: <html>
 2: <head>
 3: <title>Listing 5.14</title>
 4: </head>
 5: <body>
 6: <?php
 7: $display_prices = true;
 8: if ($display_prices) {
 9: ?>
10:     <table border="1">
11:     <tr><td colspan="3">today's prices in dollars</td></tr>
12:     <tr><td>14</td><td>32</td><td>71</td>
13:     </table>
14: <?php
15: }
16: ?>
17: </body>
18: </html>
```

Cel mai important lucru de reţinut aici este că trecerea în modul HTML de pe linia 9 are loc doar dacă expresia condiţională a instrucţiunii `if` are valoarea `true`. Acest lucru ne elimină şi problema necesităţii caracterelor de scăpare pentru evitarea procesării ghilimelelor, precum şi de transmiterea datelor către browser cu ajutorul instrucţiunii `echo` sau al funcţiei `print()`. Pe de altă parte, această abordare ar putea face ca scripturile dumneavoastră să fie mai greu de citit, mai ales atunci când veţi începe să scrieţi scripturi din ce în ce mai mari.

Rezumat

În acest capitol, aţi învăţat despre structurile de control şi despre modalităţi prin care ele vă ajută să scrieţi scripturi flexibile şi dinamice. Cele mai multe dintre aceste instrucţiuni vor reapărea în mod regulat pe parcursul acestei cărţi.

Aţi învăţat cum să definiţi o instrucţiune `if` şi cum să oferiţi variante alternative de acţiune cu ajutorul clauzelor `else` şi `else if`. Aţi învăţat să folosiţi instrucţiunea `switch` pentru a schimba cursul execuţiei în funcţie de mai multe teste de

echivalență asupra rezultatului unei singure expresii. Ați învățat despre structuri repetitive — în speță, despre instrucțiunile `while` și `for` — și ați învățat cum să folosiți instrucțiunea `break` și instrucțiunea `continue` pentru a opri prematur execuția unei asemenea structuri repetitive și, respectiv, execuția unei iterații. La final, v-au fost arătate câteva tehnici de folosire a marcajelor de deschidere și închidere a codului PHP în conjuncție cu blocurile de cod ale instrucțiunilor condiționale.

Acum ar trebui să știți suficiente elemente de bază pentru a putea scrie singur scripturi care să ia decizii și să efectueze sarcini repetitive. În capitolul următor, vom discuta despre o modalitate prin care puteți să adăugați și mai multă putere aplicațiilor dumneavoastră. Veți afla cum vă permit funcțiile să vă organizați codul, prevenind scrierea de cod redundant și crescând reutilizabilitatea codului.

Întrebări şi răspunsuri

Î. Este obligatoriu ca rezultatul unei expresii condiţionale folosite într-o structură de control să fie o valoare booleană?

R. Până la urmă, da, dar, în contextul unei expresii condiţionale, valoarea zero, o variabilă nedefinită sau un şir de caractere vid vor fi convertite la valoarea `false`. Toate celelalte valori vor fi evaluate la valoarea `true`.

Î. Trebuie să pun întotdeauna blocul de cod al unei instrucţiuni de control între acolade?

R. Dacă fragmentul de cod care trebuie executat ca parte a unei instrucţiuni condiţionale este format dintr-o singură linie de cod, puteţi omite acoladele. Cu toate acestea, obiceiul de a pune întotdeauna acolade de deschidere şi de închidere, indiferent de lungimea structurii de cod, este un obicei cât se poate de bun.

Atelier

Atelierul este conceput să anticipeze posibile întrebări, să vă ajute să recapitulaţi cele învăţate şi să începeţi să vă puneţi cunoştinţele în practică.

Test

1. Cum aţi folosi o instrucţiune `if` pentru a afişa un şir de caractere, `"Youth message"` în browser dacă o variabilă de tip integer, `$age`, este între 18 şi 35? Dacă `$age` conţine orice altă valoare, în browser ar trebui afişat şirul de caractere `"Generic message"`.
2. Cum aţi face să extindeţi codul de la întrebarea 1 astfel încât şirul `"Child message"` să fie afişat dacă valoarea variabilei `$age` este cuprinsă între 1 şi 17?
3. Cum aţi crea o instrucţiune `while` care să parcurgă şi să afişeze în browser toate numerele impare cuprinse între numerele 1 şi 49?
4. Cum aţi face pentru a converti instrucţiunea `while` pe care aţi creat-o pentru întrebarea 3 într-o instrucţiune `for`?

Soluții

1.

```
$age = 22;

if (($age >= 18) && ($age <= 35)) {
    echo "Youth message<br>\n";
} else {
    echo "Generic message<br>\n";
}
```

2.

```
$age = 12;

if (($age >= 18) && ($age <= 35)) {
    echo "Youth message<br>\n";
} elseif (($age >= 1) && ($age <= 17)) {
    echo "Child message<br>\n";
} else {
    echo "Generic message<br>\n";
}
```

3.

```
$num = 1;

while ($num <= 49) {
    echo "$num<br>\n";
    $num += 2;
}
```

4.

```
for ($num = 1; $num <= 49; $num += 2) {
    echo "$num<br>\n";
}
```

Activitate

Recapitulaţi sintaxa structurilor de control. Gândiţi-vă în ce fel vă vor putea ajuta tehnicile pe care le-aţi învăţat atunci când vă veţi scrie propriile scripturi. Poate că unele dintre ideile de scripturi pe care le dezvoltaţi ar putea să se comporte în moduri diferite în funcţie de datele introduse de utilizator, sau ar putea include bucle iterative care să afişeze un tabel HTML. Începeţi să construiţi structuri de control pe care le veţi folosi. Deocamdată, folosiţi variabile temporare pentru a simula datele introduse de utilizator sau provenite din interogări asupra unor baze de date.

CAPITOLUL 6

Folosirea funcțiilor

Funcţiile stau la baza unui script bine organizat şi permit codului să fie uşor de folosit şi reutilizat. Nici un proiect mare nu ar putea fi gestionat fără ajutorul funcţiilor. De-a lungul acestui capitol, vom discuta despre funcţii şi vom demonstra câteva dintre modalităţile prin care ele vă pot scăpa de munca redundantă.

În acest capitol, veţi învăţa cum să:

- definiţi şi să apelaţi funcţii;
- pasaţi valori funcţiilor şi să primiţi valorile returnate de acestea;
- apelaţi o funcţie în mod dinamic folosind un şir de caractere stocat într-o variabilă;
- accesaţi, din interiorul unei funcţii, variabile globale;
- creaţi funcţii „cu memorie";
- pasaţi prin referinţă valori unei funcţii;
- creaţi funcţii anonime;
- verificaţi dacă o funcţie există înainte de a o apela.

Ce este o funcție?

Puteţi să vă gândiţi la o funcţie ca la o maşină-unealtă. Maşina preia materia primă care îi este dată şi o prelucrează pentru a atinge un scop sau pentru a asambla un produs. O funcţie acceptă valorile pe care i le trimiteţi, le procesează şi, apoi, efectuează o acţiune (afişează ceva în browser, de exemplu), returnează o valoare, sau ambele.

Dacă aveţi de făcut o prăjitură, puteţi să o faceţi singur. Dacă, însă, trebuie să faceţi o mie de prăjituri, o să construiţi sau o să cumpăraţi o maşină de făcut prăjituri. La fel, atunci când hotărâţi dacă trebuie să creaţi o funcţie sau nu, cel mai important factor de luat în seamă este gradul în care acest lucru vă ajută să evitaţi scrierea de cod repetitiv.

O funcţie este un bloc de cod separat care poate fi apelat de către scripturile dumneavoastră. Codul funcţiei se execută atunci când funcţia este apelată. Puteţi să transmiteţi (pasaţi) funcţiei diverse valori pe care aceasta să le poată folosi. După ce îşi termină execuţia, funcţia poate să returneze o valoare codului care a apelat-o.

Apelarea funcţiilor

Funcţiile sunt de două tipuri — cele predefinite, implementate deja în limbaj şi cele definite de utilizator, pe care le scrieţi chiar dumneavoastră. PHP are sute de funcţii predefinite. Uitaţi-vă la următoarea porţiune de cod care exemplifică folosirea unei funcţii:

```
print ("Hello Web!");
```

În acest exemplu este apelată funcţia `print()` căreia i se pasează şirul de caractere `"Hello Web!"`. După aceasta, funcţia trece la treabă şi afişează şirul de caractere. Un apel către o funcţie conţine numele funcţiei (`print`, în acest caz) urmat de o pereche de paranteze. Dacă vreţi să pasaţi date către funcţie, acestea trebuie introduse în interiorul parantezelor. Datele astfel pasate funcţiei se numesc *argumente*. Unele funcţii au nevoie de mai multe argumente, iar acestea trebuie separate prin virgule:

```
o_functie($un_argument, $alt_argument);
```

Funcţia `print()` este un exemplu tipic de funcţie care returnează o valoare. Majoritatea funcţiilor vă trimit înapoi nişte date după ce îşi termină de efectuat sarcina — de obicei vă spun cel puţin dacă şi-au îndeplinit cu succes misiunea. `print()` returnează o valoare de tip `boolean`.

Apropo

> Funcţiile `print()` şi `echo()` au un efect similar şi vor fi folosite interşanjabil pe parcursul acestei cărţi. Puteţi să o folosiţi pe oricare dintre ele, după cum doriţi dumneavoastră.

Funcţia `abs()`, de exemplu, are nevoie de o valoare de tip număr cu semn şi returnează valoarea absolută a respectivului număr. Hai să încercăm să o folosim în listingul 6.1.

LISTINGUL 6.1. Apelarea funcţiei predefinite `abs()`.

```
 1: <html>
 2: <head>
 3: <title>Listing 6.1</title>
 4: </head>
 5: <body>
 6: <?php
 7: $num = -321;
 8: $newnum = abs( $num );
 9: echo $newnum;
10: //prints "321"
11: ?>
12: </body>
13: </html>
```

În acest exemplu, atribuim variabilei `$num` valoarea `-321`. Apoi, pasăm această variabilă funcţiei `abs()` care face calculele necesare şi returnează o valoare nouă. Atribuim această valoare variabilei `$newnum` şi afişăm rezultatul.

Puneţi aceste linii într-un fişier text denumit `abs.php` şi amplasaţi fişierul în directorul rădăcină al serverului dumneavoastră Web. Atunci când accesaţi acest script cu ajutorul browserului, veţi vedea următorul rezultat:

```
321
```

De fapt, am fi putut să renunţăm complet la variabilele temporare şi să pasăm direct funcţiei `abs()` numărul nostru afişând în acelaşi timp şi rezultatul:

```
echo abs(-321);
```

Am folosit, totuşi, variabilele temporare `$num` şi `$newnum` pentru a evidenţia cât mai clar cu putinţă fiecare pas al procesului. Uneori puteţi să faceţi codul mai lizibil spărgându-l într-un număr mai mare de expresii simple.

Puteţi apela o funcţie definită de utilizator exact în acelaşi fel în care am apelat funcţiile predefinite.

Definirea unei funcții

Vă puteţi defini propriile funcţii folosind instrucţiunea `function`:

```
function o_functie($argumentul_1, $argumentul_2) {
        //aici intra codul functiei
}
```

Numele funcţiei se introduce imediat după instrucţiunea `function` şi este urmat de o pereche de paranteze. Dacă funcţia are nevoie de argumente, trebuie să introduceţi numele variabilelor, separate prin virgule, între cele două paranteze. Aceste variabile vor prelua valorile ce îi sunt pasate funcţiei dumneavoastră. Chiar dacă funcţia nu are nevoie de argumente, trebuie întotdeauna să scrieţi şi parantezele.

Apropo

Regulile privind denumirea funcțiilor sunt similare cu cele referitoare la numele variabilelor pe care le-ați învățat în capitolul 4, intitulat *Elementele constitutive ale limbajului PHP*. Numele funcțiilor nu pot include spații și trebuie să înceapă cu o literă sau cu un simbol de subliniere (underscore).

În listingul 6.2 este declarată o funcţie.

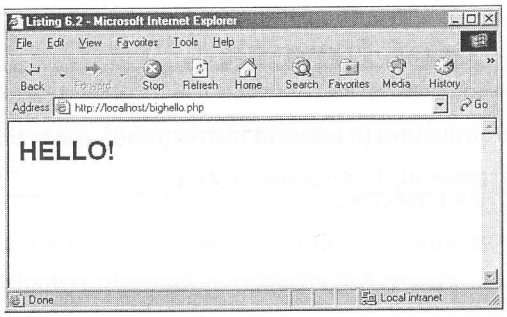
LISTINGUL 6.2. Declararea unei funcții.

```
 1: <html>
 2: <head>
 3: <title>Listing 6.2</title>
 4: </head>
 5: <body>
 6: <?php
 7: function bighello() {
 8:       echo "<h1>HELLO!</h1>";
 9: }
10: bighello();
11: ?>
12: </body>
13: </html>
```

Scriptul din listingul 6.2 are drept valoare de ieşire numai şirul de caractere `"HELLO"` încadrat între marcajele elementului `h1` al limbajului HTML.

Puneţi aceste linii într-un fişier text denumit `bighello.php` şi amplasaţi acest fişier în directorul rădăcină al serverului dumneavoastră Web. Atunci când accesaţi acest script cu ajutorul browserului, rezultatul ar trebui să arate ca în figura 6.1.

FIGURA 6.1. Rezultatul scriptului din listingul 6.2.



Am declarat o funcţie, `bighello()`, care nu are nevoie de nici un argument. De aceea, nu am scris nimic între paranteze. Deşi funcţia `bighello()` funcţionează, ea nu este extrem de utilă. În listingul 6.3 este creată o funcţie care necesită un argument şi care chiar face ceva cu acesta.

LISTINGUL 6.3. Declararea unei funcții care necesită un argument.

```
1: <html>
2: <head>
3: <title>Listing 6.3</title>
4: </head>
5: <body>
6: <?php
7: function printBR($txt) {
```

LISTINGUL 6.3. (*continuare*)

```
 8:      echo "$txt<br> \n";
 9: }
10: printBR("This is a line");
11: printBR("This is a new line");
12: printBR("This is yet another line");
13: ?>
14: </body>
15: </html>
```

Apropo

Spre deosebire de numele variabilelor, numele funcțiilor nu depind de capitalizare. În exemplul de mai sus, funcția `printBR()` ar fi putut fi apelată cu succes şi sub forma `printbr()`, `PRINTBR()` sau orice altă combinație de acest fel.

Puneţi aceste linii într-un fişier text denumit `printbr.php` şi amplasaţi acest fişier în directorul rădăcină al serverului dumneavoastră Web. Atunci când accesaţi acest script cu ajutorul browserului, rezultatul ar trebui să arate ca în figura 6.2.

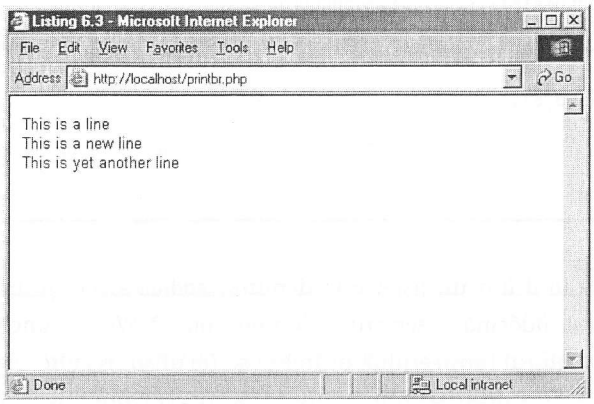


FIGURA 6.2. O funcție care afişează un şir de caractere adăugându-i marcajul <br>.

În linia 7, funcţia `printBR()` aşteaptă un şir de caractere, aşa că introducem un nume de variabilă, `$txt`, între paranteze atunci când declarăm funcţia. Datele care îi sunt pasate funcţiei `printBR()` vor fi stocate în variabila `$txt`. În interiorul corpului funcţiei, pe linia 8, afişăm variabila `$txt`, adăugându-i un element `<br>` şi un caracter de rând nou.

Ori de câte ori vrem să scriem un rând în browser, aşa cum se întâmplă în liniile 10,11 şi 12, putem să apelăm funcţia `printBR()` în locul funcţiei predefinite `print()`, scăpând astfel de nevoia de a tasta de fiecare dată elementul `<br>`.

Valorile returnate de către funcțiile definite de utilizator

În exemplul precedent, funcția `printBR()` afișează în browser un șir de caractere modificat. Uneori, totuși, veți dori ca funcția dumneavoastră să vă ofere o valoare pe care să o folosiți dumneavoastră, nu să fie afișată în browser. Dacă funcția transformă un șir de caractere pe care i l-ați furnizat, ați putea dori ca, apoi, să preluați șirul de caractere modificat pentru a-l putea pasa către alte funcții. O funcție poate returna o valoare prin intermediul instrucțiunii `return`, folosită împreună cu valoarea respectivă. Instrucțiunea `return` oprește execuția funcției și trimite valoarea înapoi către codul care a apelat funcția.

În listingul 6.4 este creată o funcție ce returnează suma a două numere.

LISTINGUL 6.4. O funcție care returnează o valoare.

```
1: <html>
2: <head>
3: <title>Listing 6.4</title>
4: </head>
5: <body>
6: <?php
7: function addNums($firstnum, $secondnum) {
8:       $result = $firstnum + $secondnum;
9:       return $result;
10: }
11: echo addNums(3,5);
12: //va afisa "8"
13: ?>
14: </body>
15: </html>
```

Puneți aceste linii într-un fișier text denumit `addnums.php` și amplasați acest fișier în directorul rădăcină al serverului dumneavoastră Web. Atunci când accesați acest script cu ajutorul browserului, ar trebui să fie afișat rezultatul următor:

```
8
```

Observați că pe linia 7, funcția `addNums()` trebuie apelată cu două argumente numerice (în linia 11 din exemplu acestea sunt 3 și 5). Argumentele sunt stocate în variabilele `$firstnum` și `$secondnum`. După cum este de așteptat, funcția `addNums()` adună numerele conținute în aceste variabile și stochează rezultatul într-o variabilă denumită `$result`.

Instrucțiunea `return` poate returna o valoare sau absolut nimic. Valoarea pe care o returnează instrucțiunea `return` poate fi obținută în mai multe moduri. Valoarea poate fi codată static:

```
return 4;
```

Valoarea poate fi rezultatul unei expresii:

```
return $a/$b;
```

În fine, valoarea poate fi rezultatul returnat de un alt apel către o funcţie:

```
return o_alta_functie($un_argument);
```

Domeniul de definire al variabilelor

O variabilă declarată în interiorul unei funcţii rămâne locală la nivelul respectivei funcţii. Cu alte cuvinte, ea nu va fi disponibilă din afara funcţiei sau din interiorul altei funcţii. În cazul proiectelor mari, acest lucru previne suprascrierea accidentală a unei variabile atunci când declaraţi două variabile cu acelaşi nume în funcţii diferite.

În listingul 6.5 este creată o variabilă în interiorul unei funcţii şi apoi se încearcă afişarea valorii acesteia din afara funcţiei.

LISTINGUL 6.5. Domeniul de definire al variabilelor: o variabilă declarată în interiorul unei funcții nu este disponibilă din afara funcției respective.

```
 1: <html>
 2: <head>
 3: <title>Listing 6.5</title>
 4: </head>
 5: <body>
 6: <?php
 7: function test() {
 8:      $testvariable = "this is a test variable";
 9: }
10: echo "test variable: $testvariable<br>";
11: ?>
12: </body>
13: </html>
```

Puneţi aceste linii într-un fişier text denumit `scopetest.php` şi amplasaţi acest fişier în directorul rădăcină al serverului dumneavoastră Web. Atunci când accesaţi acest script cu ajutorul browserului, rezultatul ar trebui să arate ca în figura 6.3.

Valoarea variabilei `$testvariable` nu este afişată din cauză că în afara funcţiei `test()` nu există o astfel de variabilă. Reţineţi că tentativa din linia 10 de accesare a unei variabile inexistente nu provoacă o eroare.

În mod similar, o variabilă declarată undeva în afara unei funcţii nu va fi disponibilă în mod automat pentru codul din interiorul respectivei funcţii.

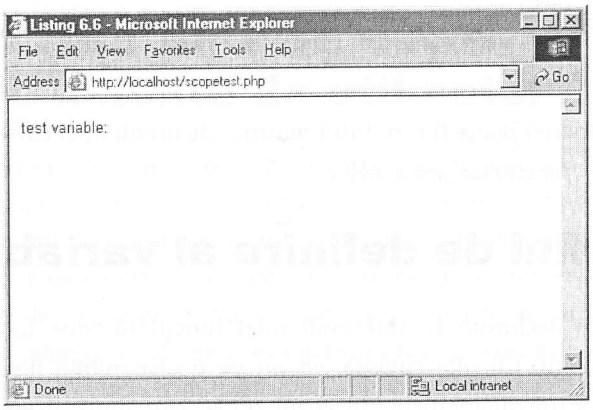


FIGURA 6.3. Rezultatul afişat de listingul 6.5.

Accesarea variabilelor cu instrucţiunea `global`

Din interiorul unei funcţii nu aveţi (în mod implicit) acces la o variabilă definită în cadrul altei funcţii sau altundeva în script. Din interiorul unei funcţii, dacă încercaţi să folosiţi o variabilă cu acelaşi nume, rezultatul va fi acela că veţi defini sau accesa o variabilă locală. Hai să testăm acest lucru în listingul 6.6.

LISTINGUL 6.6. Variabilele definite în exteriorul funcţiilor sunt, implicit, inaccesibile funcţiilor.

```
 1: <html>
 2: <head>
 3: <title>Listing 6.6</title>
 4: </head>
 5: <body>
 6: <?php
 7: $life = 42;
 8: function meaningOfLife() {
 9:      echo "The meaning of life is $life<br>";
10: }
11: meaningOfLife();
12: ?>
13: </body>
14: </html>
```

Puneţi aceste linii într-un fişier text denumit `scopetest2.php` şi amplasaţi acest fişier în directorul rădăcină al serverului dumneavoastră Web. Atunci când accesaţi acest script cu ajutorul browserului, rezultatul ar trebui să arate ca în figura 6.4.

După cum vă aşteptaţi, funcţia `meaningOfLife()` nu are acces la variabila `$life` definită pe linia 7; `$life` este vidă atunci când funcţia încearcă să o afişeze. În general, acesta este un lucru bun; ne scapă de conflictele potenţiale între variabile cu nume identice, iar o funcţie poate oricând să primească date prin intermediul

argumentelor dacă are nevoie de informaţii despre lumea din jur. Uneori, s-ar putea să aveţi, însă, nevoie să accesaţi din interiorul unei funcţii o variabilă fără să o trimiteţi sub formă de argument. În această situaţie, intră în joc instrucţiunea `global`. În listingul 6.7, instrucţiunea `global` este folosită pentru restabilirea ordinii universale.

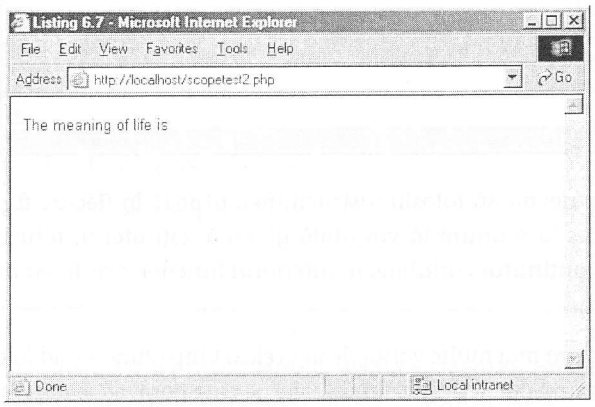


FIGURA 6.4. Încercarea de a folosi o variabilă din afara domeniului de definiție al unei funcții.

LISTINGUL 6.7. Accesarea variabilelor globale cu ajutorul instrucțiunii `global`.

```
1: <html>
2: <head>
3: <title>Listing 6.7</title>
4: </head>
5: <body>
6: <?php
7: $life=42;
8: function meaningOfLife() {
9:      global $life;
10:     echo "The meaning of life is $life<br>";
11: }
12: meaningOfLife();
13: ?>
14: </body>
15: </html>
```

Puneţi aceste linii într-un fişier text denumit `scopetest3.php` şi amplasaţi acest fişier în directorul rădăcină al serverului dumneavoastră Web. Atunci când accesaţi acest script cu ajutorul browserului, rezultatul ar trebui să arate ca în figura 6.5.

Fiindcă am pus cuvântul cheie `global` în faţa variabilei `$life` atunci am când decla-rat-o în cadrul funcţiei `meaningOfLife()`, pe linia 9, această variabilă se referă acum la variabila `$life` declarată în exteriorul funcţiei, pe linia 7.

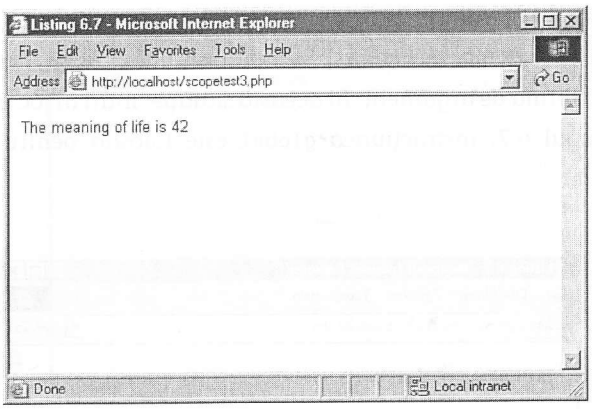


FIGURA 6.5. Accesarea reuşită a unei variabile globale din interiorul unei funcţii prin cuvântul cheie `global`.

Veţi avea nevoie să folosiţi instrucţiunea `global` în fiecare funcţie care trebuie să aibă acces la o anumită variabilă globală. Fiţi atenţi, totuşi, pentru că dacă modificaţi conţinutul variabilei în interiorul funcţiei, acesta va fi schimbat pentru întregul script.

Puteţi declara mai multe variabile în acelaşi timp atunci când folosiţi instrucţiunea `global` separând pur şi simplu variabilele pe care doriţi să le accesaţi prin virgule:

```
global $var1, $var2, $var3;
```

Atenţie!

> De obicei, un argument este o copie a datelor trimise de către codul apelant; dacă schimbaţi un argument în interiorul unei funcţii, acest lucru nu are efect în afara blocului de cod al funcţiei. Atunci când modificaţi o variabilă globală în interiorul unei funcţii, însă, acţionaţi asupra originalului, nu asupra unei copii. Folosiţi instrucţiunea `global` cu grijă.

Păstrarea stării între apelările unei funcţii prin folosirea instrucţiunii `static`

Variabilele din interiorul unei funcţii au, de obicei, o viaţă scurtă şi veselă. Ele vin pe lume atunci când funcţia este apelată şi mor atunci când execuţia funcţiei se termină, aşa cum se cuvine. De obicei, este bine să scrieţi un script sub forma unei serii de blocuri de cod separate, fiecare dintre ele având cât mai puţine informaţii despre celelalte. Uneori, totuşi, aţi putea să vreţi să dotaţi o funcţie cu o formă rudimentară de memorie.

Să presupunem că vrem să scriem o funcţie care să numere de câte ori a fost apelată astfel încât să scriptul să poată genera titluri numerotate. Am putea, evident, să folosim instrucţiunea `global` pentru a face acest lucru, după cum se poate vedea în listingul 6.8.

LISTINGUL 6.8. Folosirea instrucțiunii `global` pentru memorarea valorii unei variabile între apelările unei funcții.

```
 1: <html>
 2: <head>
 3: <title>Listing 6.8</title>
 4: </head>
 5: <body>
 6: <?php
 7: $num_of_calls = 0;
 8: function numberedHeading($txt) {
 9:     global $num_of_calls;
10:     $num_of_calls++;
11:     echo "<h1>$num_of_calls. $txt</h1>";
12: }
13: numberedHeading("Widgets");
14: echo "<p>We build a fine range of widgets</p>";
15: numberedHeading("Doodads");
16: echo "<p>Finest in the world</p>";
17: ?>
18: </body>
19: </html>
```

Puneți aceste linii într-un fișier text denumit `numberedheading.php` și amplasați fișierul în directorul rădăcină al serverului dumneavoastră Web. Atunci când accesați acest script cu ajutorul browserului, rezultatul ar trebui să arate ca în figura 6.6.

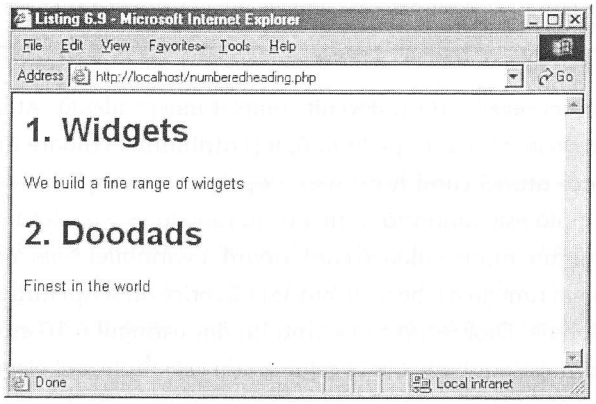


FIGURA 6.6. Folosirea instrucțiunii `global` pentru a număra de câte ori a fost apelată o funcție.

Putem să ne facem treaba în acest fel. Declarăm o variabilă, `$num_of_calls`, pe linia 7, în exteriorul funcției `numberedHeading()`. Permitem accesul din interiorul funcției la această variabilă folosind instrucțiunea `global` pe linia 9.

De fiecare dată când funcția `numberedHeading()` este apelată, valoarea variabilei `$num_of_calls` este incrementată (pe linia 10). Putem, apoi, să afișăm un titlu împreună cu numărul său de ordine.

Aceasta, însă, nu este cea mai elegantă soluţie. Funcţiile care folosesc instrucţiunea `global` nu pot fi privite ca fiind blocuri de cod independente. Atunci când le citim sau le reutilizăm, trebuie să căutăm şi variabilele globale pe care ele le manevrează.

În acest caz, poate fi utilă instrucţiunea `static`. Dacă declaraţi o variabilă în interiorul unei funcţii folosind instrucţiunea `static`, variabila rămâne locală la nivelul funcţiei, dar funcţia „ţine minte" valoarea variabilei de la o execuţie la alta. Listingul 6.9 este o adaptare a codului din listingul 6.8 şi foloseşte instrucţiunea `static`.

LISTINGUL 6.9. Folosirea instrucţiunii `static` pentru memorarea valorii unei variabile între apelările unei funcţii.

```
 1: <html>
 2: <head>
 3: <title>Listing 6.9</title>
 4: </head>
 5: <body>
 6: <?php
 7: function numberedHeading($txt) {
 8:       static $num_of_calls = 0;
 9:       $num_of_calls++;
10:       echo "<h1>$num_of_calls. $txt</h1>";
11: }
12: numberedHeading("Widgets");
13: echo "<p>We build a fine range of widgets</p>";
14: numberedHeading("Doodads");
15: echo "<p>Finest in the world</p>";
16: ?>
17: </body>
18: </html>
```

Funcţia `numberedHeading()` a devenit complet independentă. Atunci când declarăm variabila `$num_of_calls` pe linia 8, îi şi atribuim o valoare iniţială. Această atribuire se face atunci când funcţia este apelată pentru prima oară pe lina 12. Atribuirea iniţială este ignorată atunci când funcţia este apelată a doua oară pe linia 14. Codul ţine minte valoarea anterioară a variabilei `$num_of_calls`. Acum putem să copiem funcţia `numberedHeading()` în orice alt script fără să ne intereseze variabilele globale. Deşi rezultatul scriptului din listingul 6.10 este identic cu cel al scriptului din listingul 6.9, codul celui dintâi este mult mai elegant.

Mai multe despre argumente

Aţi aflat deja cum să pasaţi argumente către o funcţie, dar mai sunt multe lucruri de discutat. În această secţiune vom analiza o tehnică prin care să atribuiţi argumentelor dumneavoastră valori implicite şi vom explora o metodă prin care să pasăm variabilele prin referinţă şi nu prin valoare. Acest lucru înseamnă că funcţia primeşte un alias al valorii respective şi nu o copie a acesteia.

Stabilirea valorilor implicite ale argumentelor

PHP vă oferă o facilitate interesantă pentru construirea de funcţii flexibile. Până acum, am spus că unele funcţii necesită unul sau mai multe argumente. Făcând ca unele argumente să fie opţionale, puteţi realiza funcţii mai puţin autocratice.

Listingul 6.10 creează o funcţie micuţă dar utilă care încadrează un şir de caractere între marcajele elementului `font` al limbajului HTML. Dorim să îi oferim utilizatorului posibilitatea de a schimba atributul `size` (dimensiune) al elementului `font`, aşa că trebuie să îi cerem acestuia să furnizeze un argument `$size` pe lângă şirul de caractere (linia 7).

LISTINGUL 6.10. O funcție care necesită două argumente.

```
 1: <html>
 2: <head>
 3: <title>Listing 6.10</title>
 4: </head>
 5: <body>
 6: <?php
 7: function fontWrap($txt, $size) {
 8:     echo "<font size=\"$size\"
 9:         face=\"Helvetica, Arial, Sans-Serif\">
10:         $txt</font>";
11: }
12: fontWrap("A heading<br>",5);
13: fontWrap("some body text<br>",3);
14: fontWrap("some more body text<br>",3);
15: fontWrap("yet more body text<br>",3);
16: ?>
17: </body>
18: </html>
```

Puneţi aceste linii într-un fişier text denumit `fontwrap.php` şi amplasaţi acest fişier în directorul rădăcină al serverului dumneavoastră Web. Atunci când accesaţi acest script cu ajutorul browserului, rezultatul ar trebui să arate ca în figura 6.7.

Deşi această funcţie este foarte utilă, nu o să vrem să schimbăm dimensiunea fontului decât din când în când. De cele mai multe ori, vom dori să folosim valoarea implicită 3. Atribuind o valoare variabilei care preia argumentul în paranteza din definiţia funcţiei, putem face ca argumentul `$size` să fie opţional. Dacă apelul către funcţie nu conţine acest argument, este folosită valoarea implicită pe care i-am atribuit-o argumentului. În listingul 6.11 este folosită această tehnică pentru a face ca argumentul `$size` să fie opţional.

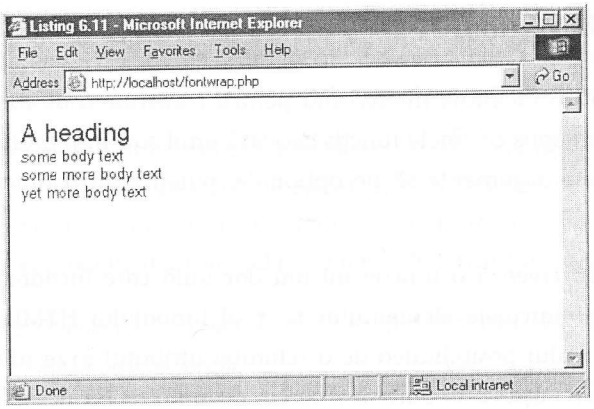


FIGURA 6.7. O funcție care formatează și afișează șiruri de caractere.

LISTINGUL 6.11. O funcție cu un argument opțional.

```
 1: <html>
 2: <head>
 3: <title>Listing 6.11</title>
 4: </head>
 5: <body>
 6: <?php
 7: function fontWrap($txt, $size = 3) {
 8:     echo "<font size=\"$size\"
 9:         face=\"Helvetica, Arial, Sans-Serif\">
10:         $txt</font>";
11: }
12: fontWrap("A heading<br>",5);
13: fontWrap("some body text<br>");
14: fontWrap("some more body text<br>");
15: fontWrap("yet more body text<br>");
16: ?>
17: </body>
18: </html>
```

Atunci când apelăm funcţia `fontWrap()` cu un al doilea argument, ca pe linia 12, această valoare este folosită pentru stabilirea atributului `size` al elementului `font`. Atunci când acest argument este omis, ca pe liniile 13, 14 şi 15, este folosită valoarea implicită 3. Puteţi crea oricâte argumente opţionale doriţi, dar o dată ce i-aţi dat unui argument o valoare implicită, trebuie să faceţi acelaşi lucru pentru toate argumentele care îi urmează.

Pasarea de referințe către funcții

Atunci când pasaţi argumente către funcţii, acestea sunt stocate ca nişte copii ale variabilelor respective. Orice schimbări aduse acestor variabile în interiorul

funcţiei sunt locale la nivelul respectivei funcţii şi nu sunt reflectate în exteriorul acesteia. Acest fapt este ilustrat în listingul 6.12.

LISTINGUL 6.12. Pasarea unui argument către o funcție prin valoare

```
 1: <html>
 2: <head>
 3: <title>Listing 6.12</title>
 4: </head>
 5: <body>
 6: <?php
 7: function addFive($num) {
 8:       $num += 5;
 9: }
10: $orignum = 10;
11: addFive($orignum);
12: echo $orignum;
13: ?>
14: </body>
15: </html>
```

Puneţi aceste linii într-un fişier text denumit `addfive.php` şi amplasaţi acest fişier în directorul rădăcină al serverului dumneavoastră Web. Atunci când accesaţi acest script cu ajutorul browserului, ar trebui să fie afişat rezultatul următor:

```
10
```

Funcţia `addFive()` acceptă o singură valoare numerică şi o adună pe aceasta cu 5. Nu returnează nimic. Atribuim o valoarea variabilei `$orignum` pe linia 10 şi, apoi, pasăm această variabilă funcţiei `addFive()` pe linia 11. O copie a conţinutului variabilei `$orignum` este stocată în variabila `$num`. Deşi incrementăm variabila `$num` cu 5, acest lucru nu are nici un efect asupra valorii variabilei `$orignum`. Când afişăm variabila `$orignum`, constatăm că valoarea ei încă este 10. În mod implicit, variabilele pasate către funcţii sunt pasate prin valoare. Cu alte cuvinte, sunt create copii locale ale valorilor variabilelor respective.

Putem schimba acest aspect creând o referinţă la variabila noastră originală. Puteţi să vă gândiţi la o referinţă ca la un indicator ce arată către o variabilă. Atunci când lucraţi cu referinţe, manevraţi valorile către care acestea arată (de multe ori, se spune că o referinţă punctează la o valoare).

Listingul 6.3 arată această tehnică în acţiune. Atunci când pasaţi prin referinţă un argument către o funcţie, aşa cum se face pe linia 11, chiar conţinutul variabilei pe care o pasaţi (`$orignum`) este accesat de către variabila argument şi manipulat în interiorul funcţiei, nu doar o copie a valorii variabilei (10). În acest caz, orice schimbări aduse argumentului se vor reflecta şi asupra valorii variabilei originale. Puteţi pasa un argument prin referinţă adăugând un *ampersand* (&) numelui argumentului în definiţia funcţiei, aşa cum vedeţi pe linia 7.

LISTINGUL 6.13. Folosirea definiției unei funcții pentru pasarea argumentelor prin referință.

```
 1: <html>
 2: <head>
 3: <title>Listing 6.13</title>
 4: </head>
 5: <body>
 6: <?php
 7: function addFive(&$num) {
 8:       $num += 5;
 9: }
10: $orignum = 10;
11: addFive($orignum);
12: echo $orignum;
13: ?>
14: </body>
15: </html>
```

Puneți aceste linii într-un fișier text denumit `addfive2.php` și amplasați acest fișier în directorul rădăcină al serverului dumneavoastră Web. Atunci când accesați acest script cu ajutorul browserului, ar trebui să fie afișat rezultatul următor:

```
15
```

Verificarea existenței unei funcții

Nu știm întotdeauna dacă o funcție există înainte de a încerca să o invocăm. Diferite *builduri* ale motorului de PHP ar putea include funcții diferite, iar dacă scrieți un script care trebuie să fie rulat pe mai multe servere, s-ar putea să aveți nevoie să verificați dacă anumite componente cheie există cu adevărat. De exemplu, ați putea dori să scrieți cod care să folosească MySQL doar dacă sunt disponibile anumite funcții legate de MySQL și, în caz contrar, să înregistreze datele pur și simplu într-un fișier text.

Puteți folosi funcția `function_exists()` pentru a verifica dacă o anumită funcție este disponibilă sau nu. `function_exists()` primește drept argument un șir de caractere ce reprezintă numele funcției căutate. Ea returnează valoarea `true` dacă funcția poate fi găsită și, altminteri, valoarea `false`.

Listingul 6.14 folosește funcția `function_exists()` și ilustrează unele dintre celelalte subiecte acoperite în cadrul acestui capitol.

LISTINGUL 6.14. Verificarea existenței unei funcții.

```
1: <html>
2: <head>
3: <title>Listing 6.14</title>
4: </head>
5: <body>
```

LISTINGUL 6.14. (*continuare*)

```
 6: <?php
 7:
 8: function tagWrap($tag, $txt, $func = "") {
 9:     if ((!empty($txt)) && (function_exists($func))) {
10:         $txt = $func($txt);
11:         return "<$tag>$txt</$tag>\n";
12:     }
13: }
14:
15: function underline($txt) {
16:     return "<u>$txt</u>";
17: }
18:
19: echo tagWrap('b', 'make me bold');
20: // va afisa <b>make me bold</b>
21:
22: echo tagWrap('i', 'underline me too', "underline");
23: // va afisa <i><u>underline me too</u></i>
24:
25: echo tagWrap('i', 'make me italic and quote me',
26:     create_function('$txt', 'return "&quot;$txt&quot;";'));
27: // va afisa <i>&quot;make me italic and quote me&quot;</i>
28: ?>
29: </body>
30: </html>
```

Definim două funcții, `tagWrap()` — pe linia 8 — și `underline()` — pe linia 15. Funcța `tagWrap()` acceptă drept argumente trei șiruri de caractere: un marcaj HTML (*tag*), textul care trebuie formatat și, opțional, numele unei funcții. Ea returnează un șir de caractere formatat. Funcția `underline()` necesită un singur argument — textul ce trebuie formatat — și returnează textul încadrat între marcajele `<u>`.

Atunci când apelăm funcția `tagWrap()` pentru prima oară, pe linia 19, îi pasăm caracterul `b` și șirul de caractere `make me bold` („scrie-mă cu aldine"). Din moment ce nu i-am pasat o valoare pentru argumentul `$func`, este folosită valoarea implicită (un șir vid). Pe linia 9, verificăm dacă variabila `$func` conține vreun caracter și, dacă nu este vidă, apelăm funcția `function_exists()` pentru a verifica dacă există o funcție cu acel nume. Bineînțeles, variabila `$func` este vidă, așa că variabila `$txt` este încadrată între marcaje `<b>` pe linia 11 și este returnat rezultatul.

Apelăm din nou funcția `tagWrap()` pe linia 22 pasându-i caracterul `'i'`, ceva text și un al treilea argument: `"underline"`. `function_exists()` găsește o funcție denumită `underline()` (linia 15), așa că apelează această funcție și îi pasează drept argument variabila `$txt` înainte să fie efectuată orice altă formatare. Rezultatul este un șir de caractere italice și subliniate.

În final, pe linia 25, apelăm `tagWrap()`, care, de data asta, încadrează textul între semne de citare.

Bineînţeles, ar fi mai rapid să adăugăm de mână semnele respective textului care trebuie să fie transformat, dar am procedat astfel pentru a ilustra ideea că `function_exists()` merge la fel de bine atât pentru funcţii anonime cât şi pentru şiruri de caractere ce reprezintă nume de funcţii.

Puneţi aceste linii într-un fişier text denumit `addnums.php` şi amplasaţi acest fişier în directorul rădăcină al serverului dumneavoastră Web. Atunci când accesaţi acest script cu ajutorul browserului, rezultatul ar trebui să arate ca în figura 6.8.

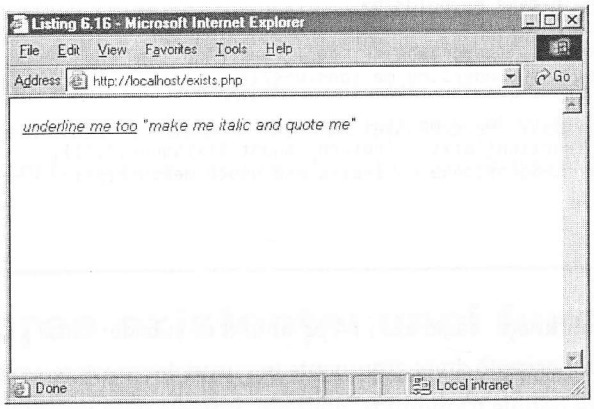


FIGURA 6.8. Rezultatul scriptului din listingul 6.14.

Rezumat

Acest capitol v-a povestit despre funcţii şi despre cum se implementează acestea. Aţi învăţat cum să definiţi şi să pasaţi argumente unei funcţii, cum să folosiţi instrucţiunile `global` şi `static`, cum să pasaţi referinţe către funcţii şi cum să creaţi valori implicite pentru argumentele funcţiilor. La final, aţi învăţat cum să verificaţi dacă o anumită funcţie există.

Întrebări şi răspunsuri

Î. Se poate include un apel către o funcţie în cadrul unui şir delimitat prin ghilimele simple sau duble, aşa cum se poate face cu o variabilă?

R. Nu. Trebuie să apelaţi funcţiile în exteriorul ghilimelelor. Totuşi, puteţi împărţi şirul de caractere şi puteţi amplasa apelul către funcţie între părţile şirului, folosind operatorul de concatenare pentru a le combina pe toate. De exemplu:

```
$newstring = "Am cumparat ".numAchizitie($un_numar)." bucati"
```

Atelier

Atelierul este conceput să anticipeze posibile întrebări, să vă ajute să recapitulaţi cele învăţate şi să începeţi să vă puneţi cunoştinţele în practică.

Test

1. Adevărat sau fals: Dacă o funcţie nu necesită nici un argument, puteţi omite complet parantezele din apelul către acea funcţie.
2. Cum returnaţi o valoare dintr-o funcţie?
3. Ce ar afişa următorul fragment de cod în browser?

```
$number = 50;

function tenTimes() {
    $number = $number * 10;
}

tenTimes();
echo $number;
```

4. Ce ar afişa următorul fragment de cod în browser?

```
$number = 50;

function tenTimes() {
        global $number;
        $number = $number * 10;
}

tenTimes();
echo $number;
```

5. Ce ar afişa următorul fragment de cod în browser?

```
$number = 50;

function tenTimes( &$n ) {
    $n = $n * 10;
}

tenTimes( $number );
echo $number;
```

Soluţii

1. Afirmaţia este falsă. Trebuie să includeţi întotdeauna parantezele în apelurile către funcţii, indiferent dacă pasaţi argumente către respectiva funcţie sau nu.
2. Trebuie să folosiţi cuvântul cheie `return`.
3. Ar afişa valoarea `50`. Funcţia `tenTimes()` nu are acces la variabila globală `$number`. Atunci când este apelată, funcţia va manipula propria ei variabilă locală `$number`.
4. Ar afişa valoarea `500`. Am folosit instrucţiunea `global`, permiţând astfel accesul funcţiei `tenTimes()` la variabila `$number`.
5. Ar afişa valoarea `500`. Adăugând caracterul ampersand (`&`) în faţa variabilei parametru `$n` ne asigurăm că acest argument este pasat prin referinţă. `$n` şi `$number` punctează la aceeaşi valoare şi, deci, orice schimbări asupra lui `$n` vor fi reflectate atunci când accesaţi variabila `$number`.

Activitate

Creaţi o funcţie care să accepte patru variabile de tip şir de caractere şi să returneze un şir de caractere care să conţină un element `table` al limbajului HTML, încadrând fiecare dintre variabile în propria ei celulă.

CAPITOLUL 7
Lucrul cu vectorii şi obiectele

Vectorii şi obiectele sunt folosite pentru stocarea şi organizarea datelor. PHP include un număr mare de funcţii care vă permit să creaţi, modificaţi şi manipulaţi vectorii. Structurile proprii programării orientate pe obiecte care se găsesc în multe limbaje de programare sunt, de asemenea, prezente în PHP.

În acest capitol, veţi învăţa chestiunile de bază ale lucrului cu vectori şi obiecte:

- cum să creaţi şi să manipulaţi vectori folosind pleiada de funcţii dedicate vectorilor incluse în PHP;
- cum să creaţi şi să manipulaţi obiectele şi datele conţinute de acestea.

Ce este un vector?

Aţi învăţat deja despre variabilele scalare în această carte, le-aţi folosit şi ştiţi că ele sunt folosite pentru stocarea valorilor. Dar variabilele scalare pot stoca numai câte o valoare în acelaşi timp. Variabila `$color` poate conţine numai o singură valoare: `red`, `blue` şi aşa mai departe — ea nu poate fi folosită pentru stocarea unei liste a culorilor curcubeului, de exemplu. Vectorii, pe de altă parte, reprezintă un tip special de variabile care vă permit să stocaţi oricâte valori doriţi.

Vectorii sunt indexaţi, ceea ce înseamnă că fiecare intrare este formată dintr-o cheie şi o valoare. Cheia reprezintă poziţia în cadrul indexului, începând de la `0`. Valoarea reprezintă orice valoare asociaţi cu respectiva poziţie — un şir de caractere, un număr întreg, sau orice altceva. Gândiţi-vă la un vector ca la un raft cu dosare. Fiecare pereche compusă din cheie şi valoare ar reprezenta, astfel, unul dintre dosarele de pe raft. Cheia ar fi eticheta scrisă pe cotorul fiecărui dosar, iar valoarea ar fi conţinutul care se află înăuntrul dosarului.

Crearea vectorilor

Puteţi crea un vector folosind fie funcţia `array()`, fie operatorul pentru vectori `[]`. Veţi folosi, de obicei, funcţia `array()` atunci când vreţi să creaţi un vector nou şi să-l

populaţi cu mai multe elemente în acelaşi timp. Veţi folosi operatorul pentru vectori atunci când veţi dori să creaţi un vector care să aibă, iniţial, un singur element sau atunci când vreţi să adăugaţi un element într-un vector deja existent.

Următoarea bucăţică de cod vă arată cum să creaţi un vector numit `$rainbow` (curcubeu), stocând în acesta toate culorile curcubeului:

```
$rainbow = array("red", "orange", "yellow", "green", "blue", "indigo", "violet");
```

Bucăţica de cod de mai jos arată crearea aceluiaşi vector în mod incremental, cu ajutorul operatorului pentru vectori:

```
$rainbow[] = "red";
$rainbow[] = "orange";
$rainbow[] = "yellow";
$rainbow[] = "green";
$rainbow[] = "blue";
$rainbow[] = "indigo";
$rainbow[] = "violet";
```

Ambele bucăţele de cod creează un vector cu şapte elemente denumit `$rainbow`, cu valori pornind de la poziţia de index 0 şi terminându-se la poziţia de index 6. Dacă aţi fi vrut să fiţi foarte exact, aţi fi putut să specificaţi poziţiile de index, ca mai jos:

```
$rainbow[0] = "red";
$rainbow[1] = "orange";
$rainbow[2] = "yellow";
$rainbow[3] = "green";
$rainbow[4] = "blue";
$rainbow[5] = "indigo";
$rainbow[6] = "violet";
```

Totuşi, PHP face acest lucru în locul dumneavoastră atunci când poziţiile nu sunt specificate şi, astfel, este eliminat riscul numerotării greşite a elementelor, precum în exemplul de mai jos:

```
$rainbow[0] = "red";
$rainbow[1] = "orange";
$rainbow[2] = "yellow";
$rainbow[5] = "green";
$rainbow[6] = "blue";
$rainbow[7] = "indigo";
$rainbow[8] = "violet";
```

Indiferent de modul în care creaţi vectorul, folosind funcţia `array()` sau operatorul pentru vectori, puteţi să îi adăugaţi elemente cu ajutorul operatorului pentru vectori:

```
$rainbow = array("red", "orange", "yellow", "green", "blue", "indigo");
$rainbow[] = "violet";
```

Exemplele folosite în această secţiune au fost vectori indexaţi numeric, posibil cel mai frecvent tip de vectori pe care îl veţi întâlni. În următoarele două secţiuni, veţi învăţa despre două alte tipuri de vectori: asociativi şi multidimensionali.

Crearea vectorilor asociativi

În vreme ce vectorii indexaţi numeric folosesc o poziţie de index pe post de cheie — 0,1,2 şi aşa mai departe — vectorii asociativi utilizează chei cu nume efective. Exemplul de mai jos ilustrează acest lucru prin crearea unui vector denumit `$character` care are patru elemente:

```
$character = array(
             "name" => "Bob",
             "occupation" => "superhero",
             "age" => 30,
             "special power" => "x-ray vision"
            );
```

Cele patru chei ale vectorului `$character` (personaj) se numesc `name`, `occupation`, `age` şi `special power` (nume, ocupaţie, vârstă şi putere specială). Valorile asociate sunt `Bob`, `superhero` (super-erou), `30` şi `x-ray vision` (vedere cu raze X). Puteţi să operaţi cu elementele specifice ale unui vector asociativ folosind cheia respectivă, ca în exemplul de mai jos:

```
<?php
echo $character['occupation'];
?>
```

Rezultatul acestui fragment de cod este:

```
superhero
```

La fel ca şi în cazul vectorilor indexaţi numeric, puteţi folosi operatorul pentru vectori ca să adăugaţi un element unui vector asociativ:

```
$character['supername'] = "Mega X-Ray Guy";
```

Acest exemplu adaugă o cheie numită `supername` (super-nume) cu valoarea `Mega X-Ray Guy`.

Crearea vectorilor multidimensionali

Primele două tipuri de vectori pot conţine şiruri de caractere şi numere întregi, pe când acest al treilea tip de vector poate conţine alţi vectori. Considerând că fiecare set de perechi cheie/valoare constituie o dimensiune, un vector multidimensional are mai multe serii de perechi cheie/valoare. De exemplu, în listingul 7.1 este definit un vector multidimensional denumit `$characters` (personaje) ale cărui elemente conţin, fiecare, câte un vector asociativ.

LISTINGUL 7.1. Definirea unui vector multidimensional.

```
1:  <?php
2:  $characters = array(
3:                    array(
4:                      "name" => "Bob",
5:                      "occupation" => "superhero",
6:                      "age" => 30,
7:                      "special power" => "x-ray vision"
8:                     ),
9:                    array(
10:                     "name" => "Sally",
11:                     "occupation" => "superhero",
12:                     "age" => 24,
13:                     "special power" => "superhuman strength"
14:                    ),
15:                    array(
16:                     "name" => "Jane",
17:                     "occupation" => "arch villain",
18:                     "age" => 45,
19:                     "special power" => "nanotechnology"
20:                    )
21:                );
22: ?>
```

Pe linia 2 este iniţializat vectorul `$characters` cu ajutorul funcţiei `array()`. Liniile 3–8 reprezintă primul element, liniile 9–14 pe cel de-al doilea, iar liniile 15–20 reprezintă cel de-al treilea element. Aceste elemente pot fi referite prin expresiile `$characters[0]`, `$characters[1]` şi `$characters[2]`.

Cu toate acestea, dacă încercaţi să afişaţi aceste elemente:

```
<?php
echo $character[2];
?>
```

rezultatul va fi următorul:

```
Array
```

Pentru a avea acces la informaţiile specifice conţinute în interiorul unui element al vectorului, trebuie să accesaţi poziţia de index a elementului şi numele asociativ al valorii pe care doriţi să o vizualizaţi. Priviţi următorul exemplu:

```
<?php
echo $character[2]['occupation'];
?>
```

Acest cod va afişa:

```
superhero
```

Câteva funcţii pentru lucrul cu vectori

În PHP sunt implementate aproximativ 60 de funcţii dedicate vectorilor, despre care puteţi afla detalii citind documentul de la `http://www.php.net/array`. În această secţiune, vor fi explicate unele dintre cele mai folosite (şi mai folositoare) astfel de funcţii.

- `count()` şi `sizeof()` — ambele numără elementele unui vector. Dat fiind următorul vector:

  ```
  $colors = array("blue", "black", "red", "green");
  ```

  atât `count($colors);` cât şi `sizeof($colors);` vor returna valoarea `4`.

- `each()` şi `list()` — acestea apar, de obicei, împreună în contextul parcurgerii unui vector element cu element şi returnării cheilor şi valorilor sale. Următorul exemplu parcurge un vector asociativ denumit `$character` afişând fiecare cheie, textul `has a value of` (are valoarea) şi valoarea, urmate de un marcaj HTML de sfârşit de linie.

  ```
  while (list($key, $val) = each($character)) {
       echo "$key has a value of $val <br>";
  }
  ```

- `foreach()` — este folosită, de asemenea, la parcurgerea unui vector, atribuind valoarea fiecărui element unei variabile. Următorul exemplu parcurge un vector asociativ denumit `$character`, afişează un text, valoarea şi, la final, marcajul HTML de sfârşit de linie.

  ```
  foreach($character as $c) {
       echo "The value is $c <br>";
  }
  ```

- `reset()` — readuce indicatorul la începutul unui vector, ca în exemplul de mai jos:

  ```
  reset($character);
  ```

  Această funcţie este utilă atunci când efectuaţi mai multe operaţii asupra unui vector, cum ar fi sortare, extragere de valori şi aşa mai departe.

- `array_push()` — adaugă unul sau mai multe elemente la sfârşitul unui vector existent, ca în acest exemplu:

  ```
  array_push($existingArray, "element 1", "element 2", "element 3");
  ```

- `array_pop()` — elimină (şi returnează) ultimul element al unui vector existent, ca în acest exemplu:

  ```
  $last_element = array_pop($existingArray);
  ```

- `array_unshift()` — adaugă unul sau mai multe elemente la începutul unui vector existent, ca în exemplul următor:

```
array_unshift($existingArray, "element 1", "element 2", "element 3");
```

- `array_shift()` — elimină (şi returnează) primul element al unui vector existent, ca în exemplul următor:

```
$first_element = array_shift($existingArray);
```

- `array_merge()` — combină doi vectori existenţi, ca în acest exemplu:

```
$newArray = array_merge($array1, $array2);
```

- `array_keys()` — returnează un vector conţinând toate numele cheilor dintr-un vector dat, ca în acest exemplu:

```
$keysArray = array_keys($existingArray);
```

- `array_values()` — returnează un vector conţinând toate valorile dintr-un vector, ca în exemplul următor:

```
$valuesArray = array_values($existingArray);
```

- `shuffle()` — randomizează elementele unui vector dat. Sintaxa acestei funcţii este pur şi simplu:

```
shuffle($existingArray);
```

Această enumerare pe scurt a funcţiilor dedicate vectorilor nu este decât o prezentare superficială a folosirii vectorilor. Totuşi, vectorii şi funcţiile de manipulare a acestora vor fi folosite de multe ori în exemplele de cod din această carte.

Crearea unui obiect

Conceptul de obiect este oarecum dificil de explicat: obiectul este un soi de cutie teoretică ce poate conţine diverse lucruri — variabile, funcţii şi aşa mai departe — şi care se bazează pe un structură ca un şablon denumită clasă. Deşi ne este destul de uşor să vizualizăm o variabilă scalară, de pildă `$color` care are valoarea `red`, sau un vector denumit `$character` cu trei sau patru elemente diferite în interior, unii dintre noi au dificultăţi atunci când trebuie să vizualizeze un obiect. Deocamdată, încercaţi să vă gândiţi la un obiect ca la o cutiuţă, în care, printr-o parte, intră date de intrare, iar, pe partea cealaltă, ies date de ieşire. Mecanismele prin care datele intră se numesc metode, iar metodele au proprietăţi. Pe parcursul acestei secţiuni, vom analiza modul în care clasele, metodele şi proprietăţile funcţionează împreună pentru a produce date de ieşire.

În deschiderea secţiunii am spus că obiectul se bazează pe o structură denumită clasă. În cadrul unei clase, sunt definite o serie de caracteristici. De exemplu, să zicem că aţi creat o clasă automobil. În cadrul clasei `automobil`, aţi putea dori să aveţi caracteristicile `culoare`, `fabricant` şi `model`. Fiecare obiect `automobil` va folosi toate aceste caracteristici, dar ele vor fi iniţializate cu valori diferite, cum ar fi `argintiu`, `Mazda` şi `Protege5` sau `roşu`, `Porsche` şi `Boxter`.

Scopul pentru care folosim obiecte este de a crea secţiuni de cod reutilizabile. Deoarece clasele sunt atât de bine structurate, complete şi independente, ele pot fi reutilizate de la o aplicaţie la alta. De exemplu, să presupunem că scrieţi o clasă de formatare a textului pentru un proiect şi vă decideţi să folosiţi respectiva clasă şi pentru un alt proiect. Fiindcă este vorba doar de un set de caracteristici, puteţi să copiaţi codul şi să îl folosiţi în cel de-al doilea proiect, operând asupra lui cu metode specifice celei de-a doua aplicaţii, dar folosind mecanismele interne ale codului existent pentru a produce rezultate noi.

Apropo

Dacă întreg conceptul de clase vă este complet străin, puteţi să aflaţi mai multe citind capitolul intitulat *Classes and Objects* din Manualul PHP. Puteţi să găsiţi acest manual la adresa `http://www.php.net/manual/ro/language.oop.php`[1].

Este destul de simplu să creaţi un obiect. Pur şi simplu declaraţi că acesta există:

```
class myClass {
        //code will go here
}
```

Acum că avem o clasă, putem crea o nouă instanţă a unui obiect:

```
$object1 = new myClass();
```

În listingul 7.2, obţinem o dovadă a existenţei obiectului nostru.

LISTINGUL 7.2. Dovada existenţei clasei.

```
1: <?php
2: class myClass {
3:       //aici va fi plasat codul
4: }
5: $object1 = new myClass();
6: echo "\$object1 is an ".gettype($object1);
7: ?>
```

Dacă salvaţi codul într-un fişier text denumit `listing7.2.php`, amplasaţi fişierul în directorul rădăcină al serverului dumneavoastră Web şi îl accesaţi cu ajutorul browserului, ar trebui să fie afişat rezultatul următor:

```
$object1 is an object
```

[1] La momentul publicării acestei cărţi, varianta în limba română a Manualului PHP nu era completă, această secţiune nefiind încă printre cele traduse (n.tr.).

Clasa noastră nu este folositoare din cauză ca nu face absolut nimic, dar este o clasă validă şi aţi putut vedea cum funcţionează modelul de lucru. În continuare, veţi învăţa despre proprietăţile şi metodele obiectelor.

Proprietăţile obiectelor

Variabilele declarate în interiorul unui obiect se numesc proprietăţi. Practica uzuală este să ca variabilele să fie declarate la începutul clasei. Aceste proprietăţi pot fi valori, vectori sau chiar alte obiecte. Fragmentul de cod care urmează foloseşte o variabilă scalară simplă în interiorul clasei, precedată de cuvântul cheie `var`:

```
class myCar {
        var $color = "silver";
        var $make = "Mazda";
        var $model = "Protege5";
}
```

Acum, de câte ori veţi crea un obiect de tip `myCar`, el va avea aceste trei proprietăţi. În listingul 7.3 vi se arată cum să accesaţi aceste proprietăţi.

LISTINGUL 7.3. Afişarea proprietăţilor unui obiect.

```
1: <?php
2: class myCar {
3:       var $color = "silver";
4:       var $make = "Mazda";
5:       var $model = "Protege5";
6: }
7: $car = new myCar();
8: echo "I drive a ".$car -> color." ".$car -> make." ".$car -> model;
9: ?>
```

Dacă salvaţi codul într-un fişier text denumit `listing7.3.php`, amplasaţi fişierul în directorul rădăcină al serverului dumneavoastră Web şi îl accesaţi cu ajutorul browserului, ar trebui să fie afişat rezultatul următor:

```
I drive a silver Mazda Protege5
```

Din moment ce probabilitatea ca şi dumneavoastră să conduceţi o Mazda Protege5 argintie sunt foarte mici, veţi dori să schimbaţi proprietăţile obiectului `myCar`. Listingul 7.4 vă arată cum să faceţi exact acest lucru.

LISTINGUL 7.4. Schimbarea proprietăţilor unui obiect.

```
1: <?php
2: class myCar {
3:       var $color = "silver";
4:       var $make = "Mazda";
5:       var $model = "Protege5";
6: }
```

LISTINGUL 7.4. *(continuare)*

```
7:  $car = new myCar();
8:  $car -> color = "red";
9:  $car -> make = "Porsche";
10: $car -> model = "Boxter";
11: echo "I drive a ".$car -> color." ".$car -> make." ".$car -> model;
12: ?>
```

Dacă salvaţi codul într-un fişier text denumit `listing7.4.php`, amplasaţi fişierul în directorul rădăcină al serverului dumneavoastră Web şi îl accesaţi cu ajutorul browserului, ar trebui să fie afişat rezultatul următor:

```
I drive a red Porsche Boxter
```

Scopul listingului 7.4 este de a vă demonstra că, în ciuda faptului că aveţi o clasă definită cu nişte proprietăţi clare, puteţi să schimbaţi valorile acestor proprietăţi cu uşurinţă în funcţie de nevoile de moment.

Metodele obiectelor

Metodele adaugă funcţionalitate obiectelor dumneavoastră. Obiectele nu vor mai sta cuminţi, ţinându-se cu dinţii de proprietăţilor lor — de acum, efectiv vor face ceva!

În listingul 7.5 vi se arată exact acest lucru:

LISTINGUL 7.5. O clasă cu o metodă.

```
1: <?php
2: class myClass {
3: function sayHello() {
4: echo "HELLO!";
5: }
6: }
7: $object1 = new myClass();
8: $object1 -> sayHello();
9: ?>
```

Deşi nu este cel mai interesant exemplu de acţiune, dacă salvaţi codul într-un fişier text denumit `listing7.5.php`, amplasaţi fişierul în directorul rădăcină al serverului dumneavoastră Web şi îl accesaţi cu ajutorul browserului, ar trebui să fie afişat rezultatul următor:

```
HELLO!
```

Iată, deci, că o metodă arată şi se comportă ca o funcţie normală, dar este definită în cadrul unei clase. Operatorul `->` este folosit pentru apelarea metodei obiectului în contextul scriptului dumneavoastră. Dacă ar fi existat variabile stocate în interiorul obiectului, metoda ar fi fost capabilă să le acceseze în cursul execuţiei sale. Acest lucru este ilustrat în listingul 7.6.

LISTINGUL 7.6. O metodă care accesează proprietățile clasei sale.

```
1:  <?php
2:  class myClass {
3:       var $name = "Matt";
4:       function sayHello() {
5:            echo "HELLO! My name is ".$this->name;
6:       }
7:  }
8:  $object1 = new myClass();
9:  $object1 -> sayHello();
10: ?>
```

Dacă salvaţi codul într-un fişier text denumit `listing7.6.php`, amplasaţi fişierul în directorul rădăcină al serverului dumneavoastră Web şi îl accesaţi cu ajutorul browserului, ar trebui să fie afişat rezultatul următor:

```
HELLO! My name is Matt
```

Folosim variabila specială `$this` pentru a ne referi la instanţa curentă a obiectului, după cum se poate vedea pe linia 5. Ori de câte ori un obiect se referă la sine însuşi, trebuie folosită variabila `$this`. Folosind variabila `$this` în conjuncţie cu operatorul `->` puteţi accesa orice proprietate sau metodă a unei clase, chiar din interiorul respectivei clase.

Un ultim amănunt de bază despre lucrul cu proprietăţile obiectelor este modul în care puteţi să schimbaţi o proprietate prin intermediul unei metode. Înainte, am schimbat o proprietate din afara clasei în care aceasta era conţinută. În listingul 7.7 vi se arată cum să faceţi acest lucru.

LISTINGUL 7.7. Schimbarea valorii unei proprietăţi din interiorul unei metode.

```
1:  <?php
2:  class myClass {
3:       var $name = "Matt";
4:       function setName($n) {
5:            $this->name = $n;
6:       }
7:       function sayHello() {
8:            echo "HELLO! My name is ".$this->name;
9:       }
10: }
11: $object1 = new myClass();
12: $object1 -> setName("Julie");
13: $object1 -> sayHello();
14: ?>
```

Dacă salvaţi codul într-un fişier text denumit `listing7.7.php`, amplasaţi fişierul în directorul rădăcină al serverului dumneavoastră Web şi îl accesaţi cu ajutorul browserului, ar trebui să fie afişat rezultatul următor:

```
HELLO! My name is Julie
```

De ce? Pentru că, pe liniile 4–6, a fost creată o nouă funcţie denumită `setName()`. Când aceasta este apelată, pe linia 12, ea schimbă valoarea proprietăţii `$name` în `Julie`. Astfel, atunci când este apelată funcţia `sayHello()`, pe linia 13, ea caută valoarea lui `$this->name` care, acum, este `Julie`, pentru că tocmai a fost modificată de funcţia `setName()`.

Cu alte cuvinte, un obiect este capabil să îşi modifice propriile proprietăţi — în cazul acesta, variabila `$name`.

Constructori

Un constructor este o funcţie care trăieşte în interiorul unei clase şi, având acelaşi nume ca şi clasa însăşi, este apelată automat ori de câte ori este creată o nouă instanţă a acelei clase prin folosirea sintaxei `new nume_clasă`. Folosirea constructorilor vă permite să furnizaţi claselor dumneavoastră argumente care să fie procesate imediat ce clasa este apelată pentru prima oară. Veţi vedea constructorii la treabă imediat, în secţiunea următoare despre moştenirea obiectelor.

Moştenirea obiectelor

Din moment ce am învăţat chestiunile de bază despre obiecte, proprietăţi şi metode, putem începe să discutăm despre moştenirea obiectelor. Moştenirea, în ceea ce priveşte clasele, înseamnă exact acest lucru: o clasă moşteneşte funcţionalitatea clasei sale părinte. Un exemplu poate fi văzut în listingul 7.8.

LISTINGUL 7.8. O clasă îşi moşteneşte părintele.

```
1:  <?php
2:  class myClass {
3:      var $name = "Matt";
4:      function myClass($n) {
5:              $this->name = $n;
6:      }
7:      function sayHello() {
8:              echo "HELLO! My name is ".$this->name;
9:      }
10: }
11: class childClass extends myClass {
12:     //aici va fi plasat codul
13: }
14: $object1 = new childClass("Baby Matt");
15: $object1 -> sayHello();
16: ?>
```

Dacă salvaţi codul într-un fişier text denumit `listing7.8.php`, amplasaţi fişierul în directorul rădăcină al serverului dumneavoastră Web şi îl accesaţi cu ajutorul browserului, ar trebui să fie afişat rezultatul următor:

```
HELLO! My name is Baby Matt
```

Liniile 4–6 reprezintă un constructor. Observaţi că funcţia are acelaşi nume ca şi clasa în care este conţinută: `myClass`. Liniile 11–13 definesc o a doua clasă, `childClass`, care nu conţine nici un fel de cod. Nu contează pentru că, în acest exemplu, am vrut doar să demonstrăm moştenirea de la clasa părinte. Moştenirea are loc datorită folosirii clauzei `extends`, pe care o puteţi vedea pe linia 11. Cea de-a doua clasă moşteneşte elementele primei clase pentru că este folosită această clauză.

Încă un exemplu, ultimul, arată cum poate o clasă copil să redefinească metodele clasei părinte — vezi listingul 7.9.

LISTINGUL 7.9. O metodă a unei clase copil ce o redefineşte pe cea a clasei părinte.

```
1: <?php
2: class myClass {
3: var $name = "Matt";
4: function myClass($n) {
5: $this->name = $n;
6: }
7: function sayHello() {
8: echo "HELLO! My name is ".$this->name;
9: }
10: }
11: class childClass extends myClass {
12: function sayHello() {
13: echo "I will not tell you my name";
14: }
15: }
16: $object1 = new childClass("Baby Matt");
17: $object1 -> sayHello();
18: ?>
```

Singurele schimbări aduse codului din listingul 7.8 sunt noile linii 12–14. Pe aceste linii, este creată o funcţie denumită `sayHello()` care, în loc să afişeze `HELLO! My name is...`, afişează mesajul `I will not tell you my name` (nu îţi voi spune numele meu). Acum, din cauză că funcţia `sayHello()` există în `childClass`, iar `childClass` este cea apelată pe linia 16, această versiune a funcţiei `sayHello()` este folosită.

Dacă salvaţi codul într-un fişier text denumit `listing7.9.php`, amplasaţi fişierul în directorul rădăcină al serverului dumneavoastră Web şi îl accesaţi cu ajutorul browserului, ar trebui să fie afişat rezultatul următor:

```
I will not tell you my name
```

La fel ca mai toate elementele programării orientate pe obiecte, moştenirea este folositoare atunci când încercăm să creăm cod flexibil. Să presupunem că aţi creat o clasă de formatare a textului care organizează şi stochează datele, în format HTML, pentru ca apoi să le afişeze în browser — o adevărată capodoperă

personală. Acum, să presupunem că aveţi un client care doreşte să folosiţi acest concept, dar, în loc să formataţi conţinutul în HTML şi să îl trimiteţi către browser, el vrea să îl formataţi în text simplu şi să îl salvaţi într-un fişier text. Nici o problemă — nu trebuie decât să adăugaţi câteva metode şi proprietăţi şi totul este gata. Până la urmă, clientul vine înapoi şi spune că are nevoie să trimită datele şi sub formă de e-mail şi, de ce nu, vrea să mai creaţi şi fişiere formatate în XML.

Deşi vă vine să vă smulgeţi părul din cap de frustrare, nu vă aflaţi într-o situaţie chiar aşa de rea. Dacă faceţi clase separate, una pentru compilare şi stocare şi încă vreo câteva pentru formatare — câte una pentru fiecare metodă de codificare (HTML, text, e-mail, XML) — aveţi, practic, o relaţie părinte–copil. Consideraţi-o drept clasă părinte pe aceea în care se găsesc metodele de compilare şi stocare. Clasele de formatare sunt clase copil — ele moştenesc informaţiile clasei părinte şi produc rezultate cu ajutorul funcţionalităţii lor proprii.

Rezumat

În acest capitol v-au fost prezentate conceptele de vector şi obiect. Mai întâi, aţi învăţat despre câteva tipuri diferite de vectori şi despre modul în care ele sunt create şi utilizate. Aceste tipuri de vectori sunt vectorii indexaţi numeric, vectorii asociativi şi vectorii multidimensionali. În plus, aţi văzut exemple de utilizare a unora dintre numeroasele funcţii dedicate vectorilor care sunt deja implementate în PHP. Aceste funcţii pot fi folosite pentru manipularea şi modificarea vectorilor existenţi şi, uneori, pentru crearea unor vectori complet noi.

În partea a doua a acestei ore, aţi căpătat noţiunile fundamentale de lucru cu codul orientat pe obiecte. Această oră nu acoperă în nici un caz toate aspectele programării orientate pe obiecte — la facultăţile de profil sunt predate serii ample de cursuri dedicate acestui subiect, deci vă puteţi imagina că aceste pagini sunt destul de lejere. Cu toate acestea, aţi învăţat să creaţi clase şi să instanţiaţi obiecte pe baza lor. Aţi învăţat să creaţi şi să accesaţi proprietăţile şi metodele unei clase, să construiţi clase noi şi să moşteniţi caracteristici de la o clasă părinte. Credeţi-mă că nu este puţin lucru!

Întrebări şi răspunsuri

Î. *Trebuie să înţeleg programarea orientată pe obiecte pentru a deveni un bun programator de PHP, sau chiar şi pentru a putea termina această carte?*

R. Nicidecum. De fapt, mare parte din această carte nu conţine programare orientată pe obiecte (OO)! Programarea orientată pe obiecte este o abordare organizaţională intenţionată să amelioreze reutilizabilitatea şi extesibilitatea codului care compune o anumită aplicaţie. La începutul proiectului, se prea poate să nu ştiţi suficiente despre proiectul în sine pentru a îl putea planifica sub forma unui concept OO. Atunci când proiectul este finalizat — sau, cel puţin, când se apropie de o formă cât de cât coerentă — aţi putea să începeţi să depistaţi locurile în care se poate adopta o abordare OO şi aţi putea începe să vă structuraţi codul în clase, proprietăţi şi metode. Dar, în cea mai mare parte a timpului, scripturile simple care îndeplinesc sarcini precise nu trebuie scrise în modalitatea OO în afara cazului în care aveţi experienţă în domeniu şi această structură vă este mai la îndemână.

Atelier

Atelierul este conceput să anticipeze posibile întrebări, să vă ajute să recapitulaţi cele învăţate şi să începeţi să vă puneţi cunoştinţele în practică.

Test

1. Care este sintaxa folosită pentru definirea unui vector?
2. Ce funcţie aţi folosi pentru a uni doi vectori?
3. Cum aţi declara o clasă denumită `emptyClass` care să nu aibă nici o metodă şi nici o proprietate?
4. Ce nume aţi alege pentru o metodă constructor?

Soluţii

1. `array()`
2. `array_merge()`
3. Se foloseşte cuvântul cheie `class`:

```
class emptyClass {
}
```

4. Nu aveţi de ales — un constructor are acelaşi nume cu clasa în care se găseşte.

Activităţi

1. Creaţi un vector multidimensional care să conţină filme organizate în funcţie de genul lor. Acesta ar trebui să fie un vector de tip asociativ în care cheile să fie reprezentate de genuri, cum ar fi `Science Fiction`, `Actiune`, `Aventuri` şi aşa mai departe. Fiecare dintre elementele vectorului ar trebui să fie un vector care să conţină nume de filme, cum ar fi `Alien`, `Terminator 3`, `Razboiul Stelelor` şi aşa mai departe. O dată ce aţi creat aceşti vectori, parcurgeţi-i element cu element afişând numele fiecărui gen şi filmele asociate acestuia.

2. Creaţi o clasă denumită `baseCalc()` în care să fie stocate două numere pe post de proprietăţi. Apoi, creaţi o metodă `calculate()` care să afişeze numerele în browser. La final, creaţi clase denumite `addCalc()`, `subCalc()`, `mulCalc()` şi `divCalc()` care să moştenească funcţionalitatea clasei `baseCalc()` dar să redefinească metoda calculate() şi să afişeze rezultatele corespunzătoare în browser (adunare, scădere, înmulţire şi, respectiv, împărţire).

PARTEA a III-a

La lucru cu codul

Capitolul 8. Lucrul cu tipurile şir de caracter, dată şi oră 153

Capitolul 9. Folosirea formularelor 181

Capitolul 10. Lucrul cu sesiunile de utilizator 203

Capitolul 11. Manipularea fişierelor şi directoarelor 213

Capitolul 12. Lucrul cu mediul de operare al serverului 237

Capitolul 13. Manipularea imaginilor 245

CAPITOLUL 8
Lucrul cu tipurile şir de caractere, dată şi oră

World Wide Web este un mediu de lucru în care se foloseşte, în general, textul simplu. Indiferent de complexitatea conţinutului afişat pe un site Web, în spatele său se găseşte codul HTML. Nu este, deci, întâmplător că în PHP există atât de multe funcţii cu ajutorul cărora puteţi formata, analiza şi manipula şirurile de caractere. Similar, datele calendaristice şi orele sunt într-atât de importante în viaţa de zi cu zi, încât ne-am obişnuit să operăm cu ele fără să trebuiască să ne gândim prea mult. Pe de altă parte, în programare este dificil de lucrat cu nuanţele calendarului gregorian. Din fericire, PHP oferă instrumente puternice pentru calculele cu date calendaristice cu ajutorul cărora manipularea datelor devine o sarcină facilă.

În acest capitol, veţi învăţa cum să:

- formataţi şiruri de caractere;
- determinaţi lungimea unui şir de caractere;
- găsiţi un subşir din cadrul unui şir de caractere;
- împărţiţi un şir de caractere în părţi componente;
- eliminaţi spaţiul alb de la începutul şi de sfârşitul unui şir de caractere;
- înlocuiţi subşiruri;
- modificaţi capitalizarea unui şir de caractere;
- preluaţi data şi ora curente;
- obţineţi informaţii despre o anumită dată calendaristică;
- formataţi informaţiile despre datele calendaristice;
- testaţi validitatea datelor calendaristice;
- configuraţi data corectă.

Formatarea şirurilor de caractere în PHP

Până acum, nu am făcut decât să afişăm direct în browser orice şir de caractere pe care îl doream afişat. PHP ne pune la dispoziţie două funcţii care ne permit să aplicăm mai întâi un şablon, fie că este vorba despre rotunjirea numerelor de tip `double` până la un număr dat de zecimale, despre definirea alinierii în cadrul unui câmp sau despre afişarea datelor în concordanţă cu diferite sisteme de numerotare. În această secţiune, veţi analiza câteva dintre opţiunile funcţiilor `printf()` şi `sprintf()`.

Folosirea funcţiei `printf()`

Dacă aveţi puţină experienţă de programare într-un limbaj de tip C, atunci funcţia `printf()` vă este familiară. Versiunea PHP este similară, dar nu identică, cu funcţia din C. În PHP, `printf()` acceptă drept argument un şir de caractere cunoscut sub numele de şir de control al formatului. Funcţia mai acceptă şi alte argumente de diferite tipuri. Şirul de control al formatului conţine instrucţiuni despre modul în care trebuie afişate aceste argumente adiţionale. În fragmentul de cod următor, de exemplu, funcţia `printf()` este folosită pentru afişarea unui număr întreg în format zecimal:

```
printf("Acesta este numarul meu: %d", 55);
//afiseaza "Acesta este numarul meu: 55"
```

În cadrul şirului de control al formatului (primul argument) este inclus un cod special, cunoscut sub numele de specificaţie de conversie.

O specificaţie de conversie începe cu caracterul procent (%) şi defineşte modul în care trebuie tratat argumentul corespunzător al funcţiei `printf()`. Puteţi include oricâte specificatoare de conversie doriţi într-un şir de control al formatului, atâta vreme cât pasaţi acelaşi număr de argumente suplimentare funcţiei `printf()`.

Următorul fragment de cod afişează două numere folosind funcţia `printf()`:

```
printf("Primul numar: %d<br>\nAl doilea numar: %d<br>\n", 55, 66);
//Rezultat:
//Primul numar: 55
//Al doilea numar: 66
```

Prima specificaţie de conversie îi corespunde primului dintre argumentele adiţionale ale funcţiei `printf()`, care este numărul 55. Cea de-a doua specificaţie de conversie îi corespunde numărului 66. Litera `d` care urmează caracterului procent indică faptul că datele trebuie tratate drept numere întregi, în sistemul zecimal. Această parte a specificaţiei de conversie se numeşte specificator de tip.

Funcţia `printf()` şi specificatorii de tip

V-aţi întâlnit deja cu un specificator de tip, `d`, care afişează datele în format zecimal. Tabelul 8.1 enumeră specificatorii de tip disponibili:

TABELUL 8.1. Specificatori de tip.

Specificator	Descriere
d	Afişează un argument sub forma unui număr zecimal.
b	Afişează un număr întreg sub forma unui număr binar.
c	Afişează un număr întreg sub forma echivalentului său ASCII.
f	Afişează un număr întreg sub forma unui număr real în virgulă mobilă („double").
o	Afişează un număr întreg sub forma unui număr octal (în baza 8).
s	Afişează argumentul sub forma unui şir de caractere.
x	Afişează argumentul sub forma unui număr hexazecimal (în baza 16), cu litere mici.
X	Afişează argumentul sub forma unui număr hexazecimal (în baza 16), cu litere mari.

În listingul 8.1 este folosită funcţia `printf()` pentru afişarea unui singur număr cu ajutorul câtorva dintre specificatorii de tip enumeraţi în tabelul 8.1.

Observaţi că în şirul de control al formatului nu se găsesc numai specificatori de tip. Orice text pe care îl includem va fi de asemenea afişat — cuvintele zecimal, binar, double, octal, şir de caractere, hex (litere mici) şi hex (litere mari) — toate în engleză în script.

LISTINGUL 8.1. O demonstraţie a câtorva dintre specificatorii de tip.

```
 1: <html>
 2: <head>
 3: <title>Listing 8.1 Demonstrating some type specifiers</title>
 4: </head>
 5: <body>
 6: <?php
 7: $number = 543;
 8: printf("Decimal: %d<br>", $number);
 9: printf("Binary: %b<br>", $number);
10: printf("Double: %f<br>", $number);
11: printf("Octal: %o<br>", $number);
12: printf("String: %s<br>", $number);
13: printf("Hex (lower): %x<br>", $number);
14: printf("Hex (upper): %X<br>", $number);
15: ?>
16: </body>
17: </html>
```

Plasaţi aceste linii de cod într-un fişier text denumit `listing8.1.php` şi amplasaţi acest fişier în directorul rădăcină al serverului dumneavoastră Web. Atunci când accesaţi acest script cu ajutorul browserului, rezultatul ar trebui să arate ca în figura 8.1.

După cum puteţi vedea, `printf()` reprezintă o modalitate rapidă de conversie a datelor dintr-o bază de numeraţie în alta şi de afişare a rezultatelor.

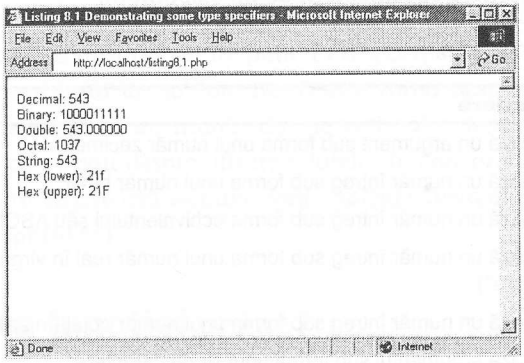


FIGURA 8.1. O demonstraţie a câtorva dintre specificatorii de tip.

Atunci când specificaţi o culoare în HTML, combinaţi trei numere hexazecimale cuprinse între `ØØ` şi `FF`, reprezentând valorile pentru roşu (`red`), verde (`green`) şi albastru (`blue`). Puteţi folosi funcţia `printf()` pentru a converti trei numere zecimale cuprinse între `Ø` şi 255 în echivalentele lor hexazecimale:

```
$red = 2Ø4
...
//afiseaza "#CCCCCC"
```

Deşi puteţi folosi specificatorii de tip pentru a converti numere zecimale în numere hexazecimale, aceştia nu vă ajută la determinarea numărului de caractere ocupate de reprezentarea fiecăruia dintre argumente. Într-un cod de culoare HTML, fiecare număr hexazecimal trebuie „lăţit" pe două caractere, ceea ce reprezintă o problemă în cazul în care am schimba variabilele noastre `$red`, `$green` şi `$blue` din fragmentul de cod precedent la valoarea `1`, de exemplu. Rezultatul ar fi, în acest caz `"#111"`. Puteţi să forţaţi reprezentarea unor zerouri în faţa numărului, folosind un specificator de completare.

Completarea rezultatului cu caractere „de umplutură" prin folosirea specificatorilor de completare

Puteţi face în aşa fel ca rezultatul să fie completat prin adăugarea unor caractere în faţa acestuia. Specificatoarele de completare trebuie să urmeze imediat după caracterul procent cu care începe o specificaţie de conversie. Pentru a completa rezultatul cu zerouri nesemnificative, specificatorul de completare trebuie să fie caracterul zero, urmat de numărul de caractere pe care doriţi să îl aibă rezultatul completat. Dacă rezultatul ocupă mai puţine caractere decât acest număr total, atunci diferenţa va fi completată cu zerouri:

```
printf("%Ø4d", 36);
//va afisa "ØØ36"
```

Pentru a completa rezultatul cu spaţii, specificatorul de completare trebuie să fie un caracter spaţiu urmat de număr de caractere pe care trebuie să le ocupe rezultatul completat:

```
printf("% 4d", 36);
//va afisa "  36"
```

Apropo

Browserul nu va afişa mai multe spaţii consecutive dintr-un document HTML. Puteţi forţa afişarea spaţiilor şi caracterelor de rând nou incluzând rezultatul între marcaje <PRE>.

```
<pre>
<?php
echo "Spatiile      vor    fi vizibile";
?>
</pre>
```

Dacă doriţi să formataţi un întreg document sub formă de text, puteţi folosi funcţia `header()` pentru a schimba antetul HTML Content-Type (tipul conţinutului).

```
header("Content-Type: text/plain");
```

Reţineţi că scriptul dumneavoastră nu trebuie să fi trimis nici un fel de date către browser pentru ca funcţia `header()` să funcţioneze corect.

Pe lângă zero şi spaţiu, puteţi specifica orice caracter cu care să fie completat rezultatul, folosind un semn de citare simplu urmat de caracterul pe care doriţi să îl folosiţi:

```
printf ("%'x4d", 36);
// afiseaza "xx36"
```

Acum avem instrumentele trebuincioase pentru a finaliza exemplul nostru de cod HTML. Până acum, puteam să convertim trei numere, dar am putea să le completăm cu zerouri:

```
$red = 1;
$green = 1;
$blue = 1;
printf("#%Ø2X%Ø2X%Ø2X", $red, $green, $blue);
//afiseaza "#Ø1Ø1Ø1"
```

Fiecare variabilă este reprezentată sub forma unui număr hexazecimal. Dacă rezultatul ocupă mai puţin de două caractere, sunt adăugate zerouri în faţa fiecărui număr.

Specificarea lăţimii unui câmp

Puteţi să stabiliţi numărul de spaţii în cadrul cărora trebuie să fie aşezat rezultatul dumneavoastră. Specificatorul de lăţime a câmpului este un număr întreg care trebuie amplasat după caracterul procent cu care începe o specificaţie de conversie (presupunând că nu este definit nici un specificator de completare).

Următorul fragment de cod afişează o listă de patru elemente aşezate fiecare în interiorul unui câmp de 20 de spaţii. Pentru ca spaţiile să fie vizibile în browser, amplasăm întreg codul în cadrul unui element PRE.

```
echo "<pre>";
printf("%20s\n", "Books");
printf("%20s\n", "CDs");
printf("%20s\n", "Games");
printf("%20s\n", "Magazines");
echo "</pre>";
```

Figura 8.2 arată rezultatul acestui fragment de cod:

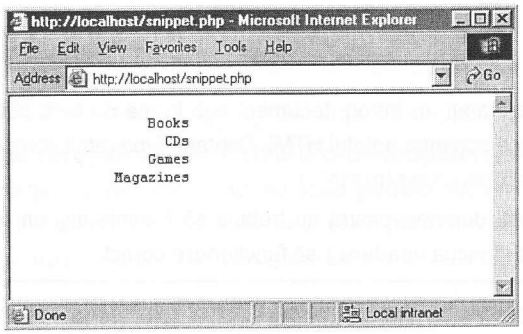


FIGURA 8.2. Alinierea la lăţimea câmpului cu ajutorul specificatorilor de lăţime.

Rezultatul este aliniat implicit la dreapta în cadrul câmpului specificat. Puteţi să îl aliniaţi la stânga adăugând un caracter minus (-) înaintea specificatorului de lăţime a câmpului:

```
printf("%-20\n", "Aliniat la stânga");
```

Reţineţi că alinierea se aplică porţiunii zecimale a oricărui număr pe care îl afişaţi. Cu alte cuvinte, numai porţiunea de dinaintea virgulei unui număr real va sta de-o parte, înspre capătul câmpului, atunci când numărul este aliniat la dreapta.

Specificarea preciziei

Atunci când doriţi să afişaţi date sub forma numerelor reale în virgulă mobilă, puteţi specifica precizia cu care trebuie rotunjite datele dumneavoastră. Acest lucru este util în special atunci când lucraţi cu sume de bani. Identificatorul de precizie trebuie amplasat imediat înainte de specificatorul de tip. El constă dintr-un punct (.) urmat de numărul de zecimale la care doriţi să se facă rotunjirea. Acest specificator are efect numai asupra datelor care sunt afişate cu specificatorul f:

```
printf("%.2f", 5.333333);
// afiseaza "5.33"
```

Apropo

În limbajul C puteţi folosi specificatorul de precizie al funcţiei `printf()` pentru a completa rezultatele zecimale cu zerouri nesemnificative. Specificatorul de precizie nu are, însă, acest efect în PHP. Folosiţi specificatorul de completare pentru a adăuga zerouri nesemnificative numerelor.

Specificaţiile de conversie: recapitulare

Tabelul 8.2 enumeră specificatorii care alcătuiesc o specificaţie de conversie în ordinea în care aceştia trebuie incluşi. Reţineţi că este dificil de folosit în acelaşi timp un specificator de completare şi unul de lăţime a câmpului. Ar trebui să alegeţi să îl folosiţi pe unul sau pe celălalt, dar nu pe amândoi în acelaşi timp.

TABELUL 8.2. Componentele specificaţiei de conversie.

Nume	Descriere	Exemplu
Specificator de completare	Determină numărul de caractere pe care ar trebui să le ocupe rezultatul şi caracterul ce trebuie adăugat în caz contrar.	`' 4'`
Specificator de lăţime a câmpului	Determină spaţiul în interiorul căruia trebuie afişat rezultatul formatat.	`'20'`
Specificator de precizie	Determină numărul de zecimale la care trebuie rotunjit un număr de tip `double`.	`'.4'`
Specificator de tip	Determină tipul de date pe care trebuie să îl aibă rezultatul.	`' d'`

În listingul 8.2 este folosită funcţia `printf()` pentru afişarea unei liste de produse şi preţuri.

LISTINGUL 8.2. Utilizarea funcţiei `printf()` pentru afişarea unei liste de produse şi preţuri.

```
 1: <html>
 2: <head>
 3: <title>Listing 8.2 Using printf() to format a list of product prices</title>
 4: </head>
 5: <body>
 6: <?php
 7: $products = array("Green armchair"=>"222.4",
 8: "Candlestick"=>"4",
 9: "Coffee table"=>80.6
10:);
11: echo "<pre>";
12: printf("%-20s%23s\n", "Name", "Price");
13: printf("%'-43s\n", "");
14: foreach ($products as $key=>$val) {
15: printf( "%-20s%20.2f\n", $key, $val );
16: }
17: echo "</pre>";
18: ?>
19: </body>
20: </html>
```

Mai întâi, pe linia 7, definim un vector asociativ care conţine numele produselor şi preţurile. Deschidem un element PRE astfel încât browserul să ne recunoască spaţiile şi caracterele de linie nouă. Primul apel la `printf()`, pe linia 12, defineşte următorul şir de control al formatului:

```
"%-20s%23s\n"
```

Prima specificaţie de conversie (`"%-20s"`) foloseşte un specificator de lăţime a câmpului de 20 de caractere, cu rezultatul aliniat la stânga. Folosim un specificator de tip şir de caractere (`"s"`). Cea de-a doua specificaţie de conversie (`"%23s"`) stabileşte un lăţimea unui câmp cu aliniere la dreapta. Acest apel la funcţia `printf()` va afişa anteturile câmpurilor.

Cel de-al doilea apel la funcţia `printf()`, de pe linia 13, desenează o linie formată din 43 de caractere „-" amplasate într-un câmp. Obţinem acest efect folosind un specificator de completare care adaugă caractere de umplutură unui şir vid.

Ultimul apel la funcţia `printf()`, de pe linia 15, face parte dintr-o instrucţiune `foreach` care parcurge vectorul nostru de produse. Folosim două specificaţii de conversie. Prima dintre ele (`"%-20s"`) afişează numele produsului sub forma unui şir de caractere aliniat la stânga în cadrul unui câmp de 20 de caractere. Cea de-a doua specificare de conversie (`"%20.2f"`) foloseşte un specificator de lăţime a câmpului pentru a se asigura că rezultatele vor fi aliniate la dreapta în cadrul unui câmp de 20 de caractere, dar şi un specificator de precizie care face ca numărul de tip `double` să fie afişat rotunjit la două zecimale.

Introduceţi aceste linii într-un fişier text denumit `listing8.2.php` şi amplasaţi acest fişier în directorul rădăcină al serverului dumneavoastră Web. Atunci când accesaţi acest script cu ajutorul browserului, rezultatul ar trebui să arate ca în figura 8.3.

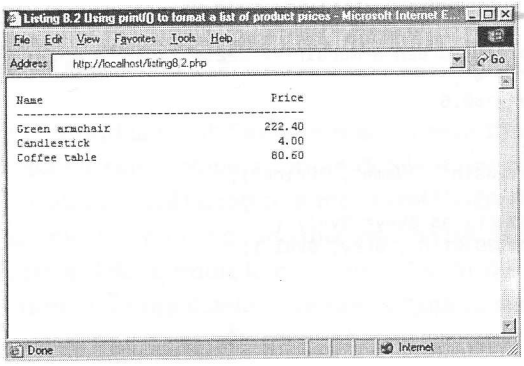


FIGURA 8.3. Produsele şi preţurile formatate cu ajutorul funcţiei `printf()`.

Schimbarea poziţiei argumentelor

Imaginaţi-vă că trebuie să afişaţi date calendaristice în browser. Aveţi datele respective într-un vector multidimensional şi folosiţi funcţia `printf()` ca să formataţi rezultatele afişate.

```
<?php
$dates = array(
            array('mon'=> 12, 'mday'=>25, 'year'=>2002),
            array('mon'=>  5, 'mday'=>23, 'year'=>2003),
            array('mon'=> 10, 'mday'=>29, 'year'=>2002)
            );

$format = include("local_format.php");

foreach($dates as $date) {
    printf("$format", $date['mon'], $date['mday'], $date['year']);
}
?>
```

În bucăţica de cod de mai sus, preluăm şirul de control al formatului dintr-un fişier inclus, `local_format.php`. Presupunând că acest fişier conţine numai:

```
<?php
return "%02d/%02d/%d<br>";
?>
```

Rezultatul nostru va avea formatul `ll/zz/aaaa` (formatul american standard pentru date calendaristice).

```
12/25/2002
05/23/2003
10/29/2002
```

Imaginaţi-vă acum că instalăm acest script pe un site britanic. În Marea Britanie, datele calendaristice sunt reprezentate cu ziua înaintea lunii (`zz/ll/aaaa`).

Codul principal nu poate fi modificat, dar fişierele de configurare, cum este şi `local_format.php`, pot fi modificate. Din fericire, acum putem schimba ordinea în care argumentele sunt aranjate în cadrul şirului de control al formatului.

```
return "%2\$02d/%1\$02d/%3\$d<br>";
```

Putem insera numărul argumentului care ne interesează înaintea caracterului procent care marchează începutul fiecărei specificări de conversie, urmat de un caracter dolar ($) protejat de procesare prin caracterul de eludare „\". Astfel, în fragmentul nostru, forţăm ca cel de-al doilea argument să fie prezentat mai întâi, urmat de primul şi, în final, de al treilea. Rezultatul este o listă de date calendaristice în format britanic.

```
25/12/2002
23/05/2003
29/10/2002
```

Stocarea unui şir de caractere formatat

Funcţia `printf()` afişează datele în browser, ceea ce înseamnă că rezultatul ei nu este disponibil pentru scripturile dumneavoastră. Puteţi, totuşi, să folosiţi funcţia `sprintf()` care funcţionează exact la fel ca `printf()`, doar că returnează un şir de caractere pe care îl puteţi, apoi, stoca într-o variabilă pentru a-l folosi mai târziu. Următorul fragment de cod foloseşte funcţia `sprintf()` pentru a rotunji un număr de tip `double` la două zecimale, stocând rezultatul într-o variabilă denumită `$dosh` (parale):

```
$dosh = sprintf("%.2f", 2.334454);
echo "Aveţi $dosh dolari de cheltuiala"
```

O modalitate specială de folosire a funcţiei `sprintf()` este reprezentată de scrierea datelor formatate într-un fişier. Puteţi apela `sprintf()` atribuind valoarea întoarsă de aceasta unei variabile pe care, apoi, o puteţi scrie într-un fişier folosind funcţia `fputs()`.

Analizarea şirurilor în PHP

Nu puteţi şti întotdeauna totul despre datele cu care lucraţi. Şirurile pot veni din multe surse, inclusiv de la utilizator, din baze de date, din fişiere sau din pagini Web. Înainte de a începe să prelucraţi date din surse externe, de multe ori, va trebui să aflaţi mai multe lucruri despre acestea. PHP vă pune la dispoziţie mai multe funcţii care vă permit să căpătaţi informaţii despre şirurile de caractere.

O notă despre indexarea şirurilor

Vom folosi destul de frecvent cuvântul index în legătură cu şirurile de caractere. Aţi auzit acest cuvânt mai ales în contextul vectorilor ca, de exemplu, în capitolul 7, „Lucrul cu vectorii şi obiectele". De fapt, şirurile şi vectorii nu sunt într-atât de diferite pe cât vi s-ar părea. Vă puteţi gândi la un şir de caractere ca la un vector de caractere. Aceasta înseamnă că dumneavoastră aveţi posibilitatea de a accesa un caracter anume dintr-un şir de caractere ca şi când ar fi un element al unui vector:

```
$test = "scallywag";
echo $test[Ø]; // afiseaza "s"
echo $test[2]; // afiseaza "a"
```

Este important de reţinut, de aceea, că atunci când vorbim despre poziţia, sau indexul, unui caracter în cadrul unui şir, caracterele, la fel ca elementele vectorilor, sunt indexate pornind de la Ø.

Aflarea lungimii unui şir cu ajutorul funcției `strlen()`

Puteţi folosi funcţia `strlen()` pentru a determina lungimea unui şir de caractere. `strlen()` acceptă drept argument un şir de caractere şi returnează un număr întreg reprezentând numărul de caractere din variabila pe care aţi pasat-o. Funcţia `strlen()` este folosită de obicei pentru verificarea lungimii textului introdus de utilizator. Următorul fragment testează un cod de identificare pentru a se asigura că are patru cifre.

```
if (strlen($membership) == 4) {
    echo "<p>Thank you!</p>";
} else {
    echo "<p>Your membership number must have 4 digits</p>";
}
```

Utilizatorului i se mulţumeşte în cazul în care variabila globală `$membership` conţine patru caractere. Altminteri, este generat un mesaj de eroare care îl anunţă că numărul de membru trebuie să aibă patru cifre.

Găsirea unui subşir în interiorul unui şir cu ajutorul funcției `strstr()`

Puteţi folosi funcţia `strstr()` pentru a verifica dacă un anumit şir de caractere există în interiorul unui alt şir de caractere. `strstr()` acceptă două argumente: un şir sursă şi subşirul pe care doriţi să îl identificaţi. Funcţia returnează valoarea `false` dacă subşirul este absent. Altminteri, funcţia returnează porţiunea din şirul sursă care începe cu subşirul căutat. Pentru următorul exemplu, imaginaţi-vă că dorim să tratăm codurile de membru care conţin şirul AB diferit faţă de cele care nu îl conţin:

```
$membership = "pAB7";
if (strstr($membership, "AB")) {
   echo "<p>Thank you. Don't forget that your membership expires soon!</p>";
} else {
   echo "<p>Thank you!</p>";
}
```

Deoarece variabila pe care o testăm, `$membership`, conţine într-adevăr şirul `AB`, `strstr()` returnează valoarea `AB7`. În cadrul expresiei condiţionale, această valoare se traduce prin `true`. Ce s-ar fi întâmplat dacă utilizatorul introducea `"pab7"`? Deoarece `strstr()` face diferenţa între literele majuscule şi literele de rând, `AB` nu ar fi fost găsit. Expresia condiţională din instrucţiunea `if` ar fi fost falsă, iar mesajul implicit ar fi fost afişat în browser. Dacă dorim să căutăm atât `AB` cât şi `ab` în interiorul şirului, trebuie să folosim funcţia `stristr()`, care operează în acelaşi fel, dar nu este dependentă de capitalizare.

Aflarea poziției unui subşir cu ajutorul funcției `strpos()`

Funcţia `strpos()` vă spune în acelaşi timp şi dacă un subşir există în interiorul unui şir de caractere mai mare, precum şi în ce poziţie se găseşte respectivul subşir.

strpos() acceptă două argumente: şirul sursă şi subşirul pe care îl căutaţi. Funcţia mai acceptă şi un al treilea argument opţional, un număr întreg care reprezintă indexul de la care trebuie să înceapă căutarea. Dacă subşirul nu există, strpos() returnează valoarea false; altminteri, funcţia returnează indexul la care începe respectivul subşir. În următorul fragment, cu ajutorul funcţiei strpos(), se verifică dacă un şir de caractere începe cu subşirul mz:

```
$membership = "mz00xyz";
if (strpos($membership, "mz") === 0) {
   echo "hello mz";
}
```

Observaţi şmecheria pe care a trebuit să o facem pentru a obţine rezultatul aşteptat. strpos() identifică subşirul mz în interiorul şirului nostru, dar el este primul element din şir de caractere. Acest lucru înseamnă că poziţia de index pe care strpos() o returnează este zero, valoarea care, în expresia noastră condiţională, s-ar traduce prin false. Pentru a evita acest rezultat, folosim operatorul de echivalenţă din PHP, ===, care întoarce valoarea true dacă ambii operanzi sunt echivalenţi şi de acelaşi tip.

Extragerea unei părţi a unui şir cu ajutorul funcţiei substr()

Funcţia substr() returnează o porţiune a unui şir de caractere pe baza indexului de pornire şi a lungimii porţiunii pe care o căutaţi. Această funcţie are nevoie de două argumente: un şir sursă şi poziţia de index la care să înceapă subşirul. Funcţia returnează toate caracterele începând cu cel din poziţia de start şi până la ultimul caracter din şir. Opţional, funcţia mai acceptă şi un al treilea argument, care ar trebui să fie un număr întreg reprezentând lungimea şirului pe care vreţi să îl returneze. Dacă acest argument este prezent, funcţia substr() returnează numai numărul de caractere specificat, începând cu respectiva poziţia de index.

```
$test = "scallywag";
echo substr($test,6);  // afiseaza "wag"
echo substr($test,6,2) // afiseaza "wa"
```

Dacă al doilea argument (indexul de pornire) pasat către substr() este un număr negativ, funcţia va număra de la sfârşitul şirului, iar nu de la începutul acestuia. Următorul fragment de cod scrie un mesaj anume pentru utilizatorii care au introdus o adresă de e-mail care se termină cu .uk:

```
$test = "matt@corrosive.co.uk";
if ($test = substr($test, -3) == ".uk") {
   echo "<p>Don't forget our special offers for British customers!</p>";
} else {
   echo "<p>Welcome to our shop!</p>";
}
```

Împărţirea unui şir în fragmente cu ajutorul funcţiei `strtok()`

Puteţi procesa un şir de caractere cuvânt cu cuvânt folosind funcţia `strtok()`. Această funcţie are nevoie iniţial de două argumente: şirul care trebuie împărţit şi elementul de delimitare pe baza căruia şirul trebuie divizat. Elementul de delimitare trebuie să fie un şir de caractere, iar acesta trebuie să aibă oricâte caractere doriţi. Funcţia returnează primul fragment astfel delimitat pe care îl găseşte în cadrul şirului sursă. După ce funcţia `strtok()` a fost apelată prima oară, şirul sursă este păstrat în memorie. La apelări ulterioare nu trebuie să îi pasaţi funcţiei decât şirul de delimitare. Funcţia va întoarce următorul fragment găsit al fiecare apel succesiv, returnând `false` atunci când ajunge la sfârşitul şirului sursă. Funcţia `strtok()` este apelată, de obicei, iterativ, în cadrul unei bucle. Listingul 8.3 evidenţiază utilizarea funcţiei `strtok()` pentru fragmentarea unui URL prin separarea adresei şi a căii, pe de-o parte, de şirul de interogare, pe de altă parte şi, apoi, prin împărţirea perechilor nume/valoare ale şirului de interogare.

LISTINGUL 8.3. Împărţirea unui şir în fragmente delimitate prin folosirea funcţiei `strtok()`.

```
 1: <html>
 2: <head>
 3: <title>Listing 8.3 Dividing a string into tokens with strtok()</title>
 4: </head>
 5: <body>
 6: <?php
 7: $test  = "http://www.deja.com/qs.xp?";
 8: $test .= "OP=dnquery.xp&ST=MS&DBS=2&QRY=developer+php";
 9: $delims = "?&";
10: $word = strtok($test, $delims);
11: while (is_string($word)) {
12:    if ($word) {
13:       echo "$word<br>";
14:     }
15:    $word = strtok($delims);
16: }
17: ?>
18: </body>
19: </html>
```

Puneţi aceste linii într-un fişier text denumit `listing8.3.php` şi amplasaţi acest fişier în directorul rădăcină al serverului dumneavoastră Web. Atunci când accesaţi acest script cu ajutorul browserului, rezultatul ar trebui să arate ca în figura 8.4.

Funcţia `strtok()` este un instrument destul de dificil de utilizat şi pentru a lucra cu el sunt necesare câteva artificii. Mai întâi stocăm delimitatorii pe care vrem să îi folosim într-o variabilă, `$delims`, pe linia 9. Apelăm `strtok()` pe linia 10, pasându-i URL-ul pe care vrem să îl fragmentăm, precum şi şirul `$delims`. Stocăm primul rezultat în variabila `$word`. În cadrul expresiei condiţionale a buclei `while` de pe linia 11, verificăm dacă `$word` este un şir. Dacă nu este, ştim că am ajuns la sfârşitul şirului sursă şi că nu mai avem nimic de făcut.

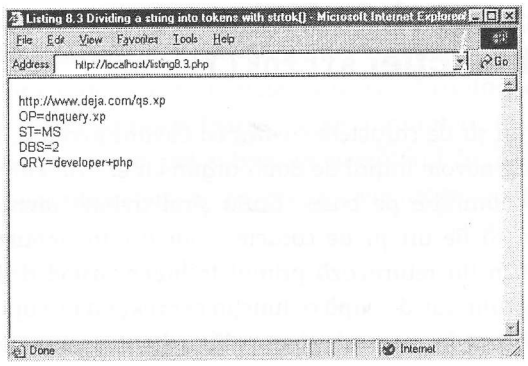


FIGURA 8.4. Rezultatul listingului 8.3, un şir fragmentat pe baza unor delimitatori.

Testăm tipul valorii returnate pentru că un şir de caractere care conţine doi sau mai mulţi delimitatori consecutivi face ca `strtok()` să întoarcă un şir vid atunci când îl întâlneşte pe primul dintre ei. De aceea, o expresie condiţională convenţională cum ar fi:

```
while ($word) {
      $word = strtok($delims);
}
```

ar fi falsă ori de câte ori `$word` ar fi un şir vid, chiar şi atunci când nu se ajunge de fapt la sfârşitul şirului sursă.

După ce am stabilit că `$word` conţine un şir de caractere, putem să lucrăm cu el. Dacă `$word` nu conţine un şir vid, îl afişăm în browser (linia 13). Trebuie să apelăm, apoi, funcţia `strtok()` din nou, pe linia 15, pentru a repopula variabila `$word` în vederea testului următor. Observaţi că nu pasăm şirul sursă către funcţia `strtok()` decât la primul apel. Dacă am face asta şi la apelurile ulterioare, funcţia ar returna de fiecare dată primul cuvânt din şirul sursă, iar noi am avea de-a face cu o buclă infinită.

Manipularea şirurilor în PHP

PHP ne pune la dispoziţie multe funcţii care primesc drept argument un şir de caractere şi îl transformă, după cum veţi vedea, fie în mod subtil, fie în mod radical.

Curăţarea unui şir cu ajutorul funcţiilor `trim()` şi `ltrim()` şi `strip_tags()`

Atunci când preluaţi un şir de caractere de la utilizator sau dintr-un fişier, nu puteţi fi întotdeauna sigur că nu aţi cules cu ocazia asta şi spaţii albe la începutul sau la sfârşitul datelor utile. Funcţia `trim()` elimină orice spaţii albe, inclusiv caracterele spaţiu, de linie nouă şi de tabulare, atât de la începutul cât şi de la

sfârşitul unui şir de caractere. Funcţia acceptă drept argument şirul care trebuie modificat şi returnează versiunea curăţată a acestuia.

```
$text = "\t\to gramada de spatiu liber              ";
echo "<pre>$text</pre>";
//afiseaza "              o gramada de spatiu liber          ";
$text = trim($text);
echo "<pre>$text</pre>";
//afiseaza "o gramada de spatiu liber      ";
```

Bineînţeles că asta ar putea fi mai mult decât aveţi nevoie. S-ar putea să doriţi să păstraţi spaţiul alb de la începutul unui şir de caractere, dar să îl eliminaţi pe cel de la sfârşit. Puteţi folosi funcţia `rtrim()` din PHP exact în acelaşi fel ca funcţia `trim()`, dar numai spaţiul alb de la sfârşitul şirului pasat ca argument va fi eliminat:

```
$text = "\t\to gramada de spatiu liber              ";
echo "<pre>$text</pre>";
//afiseaza "              o gramada de spatiu liber          ";
$text = rtrim($text);
echo $test;
//afiseaza "o gramada de spatiu liber                ";
```

PHP ne mai pune la dispoziţie şi funcţia `ltrim()` care elimină spaţiul alb numai de la începutul unui şir de caractere. La fel, funcţia este apelată pasându-i-se drept argument şirul pe care doriţi să îl transformaţi, iar ea returnează un şir nou, din care au fost scoase caracterele de tabulare, de linie nouă şi spaţiile:

```
$text = "\t\to gramada de spatiu liber              ";
echo "<pre>$text</pre>";
//afiseaza "              o gramada de spatiu liber          ";
$text = ltrim($text);
echo "<pre>$text</pre>";
//afiseaza "o gramada de spatiu liber              ";
```

Nu de puţine ori, va trebui să eliminaţi marcajele dintr-un bloc de text pentru a-l putea prezenta fără formatarea HTML. PHP ne pune la dispoziţie, în acest scop, funcţia `strip_tags()`. Această funcţie acceptă două argumente. Primul este textul care trebuie transformat. Al doilea argument este opţional şi ar trebui să fie o listă de marcaje HTML care să nu fie îndepărtate prin acţiunea funcţiei `strip_tags()`. Marcajele din lista de excepţii nu trebuie separate prin nici un caracter.

```
$string = "<p>I <i>simply</i> will not have it, <br>said Mr Dean</p> <p><b>The
end</b></p>";
echo strip_tags($string, "<br><p>");
```

În fragmentul de cod precedent am creat un şir de caractere în format HTML. Când am apelat funcţia `strip_tags()`, i-am pasat variabila `$string` şi o listă de excepţii. Rezultatul este acela că marcajele `<p>` şi `<br>` sunt lăsate la locul lor, iar toate celelalte marcaje sunt eliminate. De asemenea, marcajul de închidere al lui `<p>`—`</p>`—este la rândul său eliminat.

Înlocuirea unei porţiuni a unui şir cu ajutorul funcţiei `substr_replace()`

Funcţia `substr_replace()` operează foarte similar cu funcţia `substr()`, doar că vă permite să înlocuiţi porţiunea de şir pe care o extrageţi. Funcţia necesită trei argumente: şirul pe care îl transformaţi, textul pe care vreţi să îl adăugaţi şi indexul de start. Ea acceptă, de asemenea, un al patrulea argument opţional, lungimea subşirului. Funcţia `substr_replace()` identifică porţiunea de şir specificată prin indexul de pornire şi lungime, înlocuind această porţiune cu şirul furnizat în argumentul ce reprezintă textul de adăugat şi returnează întreg şirul transformat.

În fragmentul de cod de mai jos, pentru a reînnoi codul de membru al unui utilizator, trebuie să îi modificăm al doilea şi al treilea caracter.

```
<?php
$membership = "mzØ2xyz";
$membership = substr_replace($membership, "Ø3", 2, 2);
echo "New membership number: $membership";
// afiseaza "New membership number: mzØ3xyz"
?>
```

Rezultatul acestui cod este acela că numărul iniţial de membru, `mzØ2xyz`, devine noul număr de membru, `mzØ3xyz`.

Înlocuirea subşirurilor cu ajutorul funcţiei `str_replace()`

Funcţia `str_replace()` înlocuieşte toate apariţiile unui şir de caractere în interiorul altui şir de caractere. Funcţia necesită trei argumente: un şir de căutat, şirul cu care acesta trebuie înlocuit şi şirul sursă asupra căruia trebuie operate modificările. Funcţia returnează întreg şirul transformat. În următorul exemplu este folosită funcţia `str_replace()` pentru a înlocui toate referirile la anul 2002 din cadrul unui şir corespunzător anului 2003:

```
$string  = "Site contents copyright 2ØØ2. ";
$string .= "The 2ØØ2 Guide to All Things Good in Europe";
echo str_replace("2ØØ2","2ØØ3",$string);
```

Toate argumentele funcţiei `str_replace()` pot fi şi vectori, nu numai şiruri. Acest fapt ne permite să efectuăm operaţiuni multiple de căutare şi înlocuire atât asupra unui singur şir de caractere, cât şi asupra mai multor şiruri în acelaşi timp. Iată, de exemplu, următorul fragment de cod:

```
<?php
$source = array(
    "The package which is at version 4.2 was released in 2ØØ2",
    "The year 2ØØ2 was an excellent period for PointyThing 4.2" );
$search  = array("4.2", "2ØØ2");
$replace = array("5.Ø", "2ØØ3");
```

```
$source = str_replace($search, $replace, $source);
foreach($source as $str) {
    echo "$str<br>";
}
// afiseaza:
// The package which is at version 5.0 was released in 2003
// The year 2003 was an excellent period for PointyThing 5.0
?>
```

Atunci când primul şi al doilea argument al funcţiei `str_replace()` sunt vectori, funcţia va încerca să înlocuiască fiecare şir de căutat din vectorul care constituie primul argument, cu şirurile corespunzătoare din cel de-al doilea argument. Atunci când şi cel de-al treilea argument este un vector, funcţia va returna un vector de şiruri. În acest caz, operaţiunea de căutare şi înlocuirea vor fi fost efectuate asupra fiecărui şir din vector.

Schimbarea capitalizării literelor

PHP oferă câteva funcţii care vă permit să convertiţi literele mari şi mici dintr-un şir de caractere. Schimbarea capitalizării este de multe ori utilă pentru comparaţii între şiruri. Pentru a obţine versiunea cu litere mari a unui şir de caractere, folosiţi funcţia `strtoupper()`. Această funcţie nu are nevoie decât de un argument — şirul pe care doriţi să îl convertiţi — şi returnează şirul convertit:

```
$membership = "mz03xyz";
$membership = strtoupper($membership);
echo "$membership"; // afiseaza "MZ03XYZ"
```

Pentru a obţine un şir de caractere cu litere mici, folosiţi funcţia `strtolower()`. La fel, aceasta necesită drept argument şirul pe care doriţi să îl convertiţi şi întoarce versiunea sa convertită:

```
$membership = "MZ03XYZ";
$membership = strtolower($membership);
echo "$membership"; // afiseaza "mz03xyz"
```

PHP ne mai pune la dispoziţie şi o funcţie legată de capitalizarea literelor care este utilă din punct de vedere estetic. Funcţia `ucwords()` face în aşa fel ca prima literă a fiecărui cuvânt să fie majusculă. În fragmentul de cod următor, facem ca prima literă a fiecărui cuvânt dintr-un şir de caractere introdus de utilizator să fie literă mare:

```
$full_name = "violet elizabeth bott";
$full_name = ucwords($full_name);
echo $full_name; // afiseaza "Violet Elizabeth Bott"
```

Deşi această funcţie transformă prima literă a fiecărui cuvânt, ea nu se atinge de celelalte litere. De aceea, dacă utilizatorul a avut probleme cu tasta Shift atunci când a introdus şirul de caractere din exemplul precedent şi a scris `VIolEt eLIZaBeTH bOTt`, folosirea acestei funcţii nu a reuşit să repare aspectul şirului. Rezultatul nostru ar fi fost `VIolEt ELIZaBeTH BOTt`, ceea ce nu reprezintă o

îmbunătăţire semnificativă. Putem rezolva această problemă convertind întregul şir în litere mici cu ajutorul funcţiei `strtolower()` înainte de a folosi `ucwords()`:

```
$full_name = "VIolEt eLIZaBeTH bOTt";
$full_name = ucwords(strtolower($full_name));
echo $full_name; // afiseaza "Violet Elizabeth Bott"
```

În fine, funcţia `ucfirst()` va transforma în majuscule numai prima literă a unui şir de caractere. În fragmentul de cod următor, am transformat în majusculă prima literă a unui şir de caractere introdus de utilizator:

```
$myString = "this is my string";
$myString = ucfirst($myString);
echo $myString; // afiseaza "This is my string"
```

Ajustarea lungimii rândurilor cu ajutorul funcţiilor `wordwrap()` şi `nl2br()`

Atunci când afişaţi text simplu într-o pagină Web, de multe ori caracterele de linie nouă nu sunt afişate, iar textul curge într-o singură linie lungă şi amorfă. Funcţia `nl2br()` este o metodă comodă cu ajutorul căreia toate caracterele de linie nouă sunt convertite în marcaje HTML de sfârşit de rând. Astfel:

```
$string = "o linie\n";
$string .= "inca o linie\n";
$string .= "trei sunt cu noroc\n";
echo nl2br($string);
```

se va transforma în:

```
o linie<br />
inca o linie<br />
trei sunt cu noroc<br />
```

Observaţi că marcajele `<br>` sunt generate în format compatibil cu standardul XHTML.

Funcţia `nl2br()` este excepţională pentru valorificarea caracterelor de linie nouă care există deja în textul pe care îl convertiţi. Uneori, s-ar putea să doriţi să adăugaţi marcaje arbitrare de sfârşit de linie pentru a formata o coloană de text, iar funcţia `wordwrap()` este perfectă în acest caz. Funcţia `wordwrap()` are nevoie de un singur argument — şirul care trebuie transformat; ea ajustează implicit liniile la 75 de caractere şi foloseşte `\n` pe post de caracter de sfârşit de linie. Astfel, fragmentul de cod:

```
$string = "Data fiind o linie lunga, wordwrap() este un mijloc facil de a";
$string .= "o potrivi la dimensiunea unei coloane, facand-o mai usor de citit";
echo wordwrap($string);
```

va afişa:

```
Data fiind o linie lunga, wordwrap() este un mijloc facil de a o potrivi la
dimensiunea unei coloane, facand-o mai usor de citit
```

Din cauză că liniile sunt împărţite prin caracterul `\n`, formatarea nu va apărea, bineînţeles, în codul HTML. `wordwrap()` mai acceptă două argumente opţionale: un număr reprezentând lungimea maximă, în caractere, pe care o poate avea o linie şi un şir reprezentând secvenţa de caractere cu care doriţi să se sfârşească fiecare linie. Astfel, aplicând funcţia variabilei noastre `$string` în felul următor:

```
echo wordwrap($string, 24, "<br>\n");
```

rezultatul va fi:

```
Data fiind o linie lunga,<br>
wordwrap() este un mijloc<br>
facil de a o potriv la<br>
dimensiunea unei coloane,<br>
facand-o mai usor de <br>
citit
```

Funcţia `wordwrap()` nu va separa automat un rând dacă un cuvânt are mai multe caractere decât limita. Puteţi, totuşi, să folosiţi un al patrulea argument opţional pentru a forţa acest lucru. Argumentul trebuie să fie un număr întreg pozitiv. Astfel, folosind funcţia `wordwrap()` în conjuncţie cu cel de-al patrulea argument, putem acum să ajustăm un şir chiar dacă el conţine cuvinte care sunt mai lungi decât limita pe care o stabilim. Acest fragment de cod:

```
$string = "Ca de obicei, ma puteti gasi la http://www.witteringonaboutit.com/";
$string .= "chat/eating_green_cheese/forum.php. Sper sa ne vedem acolo!";
echo wordwrap($string, 24, "<br>\n", 1);
```

va afişa:

```
Ca de obicei ma veti gasi<br>
la<br>
http://www.witteringonab<br>
outit.com/chat/eating_gr<br>
een_cheese/forum.php.<br>
Sper sa ne vedem acolo!
```

în loc să afişeze:

```
Ca de obicei ma veti gasi<br>
la<br>
http://www.witteringonaboutit.com/chat/eating_green_cheese/forum.php. <br>
Sper sa ne vedem acolo!
```

Împărţirea şirurilor în vectori cu ajutorul funcţiei `explode()`

Funcţia cu nume extrem de sugestiv `explode()` este similară din multe puncte de vedere cu `strtok()`. Diferenţa principală este că funcţia `explode()` va stoca fragmentele şirului în elementele unui vector pe care, apoi, îl puteţi salva, sorta sau analiza după plac. Funcţia `explode()` are două argumente obligatorii: şirul delimitator pe care vreţi să îl folosiţi pentru fragmentarea şirului sursă şi, evident, şirul sursă. Funcţia mai poate accepta şi un al treilea argument opţional care stabileşte numărul maxim de fragmente în care poate fi împărţit şirul. Şirul delimitator poate include mai multe caractere şi toate, împreună, vor fi considerate elementul de delimitare (spre deosebire de modul în care, atunci când funcţiei `strtok()` i se pasează mai multe caractere, fiecare dintre acestea este, separat, un delimitator). Fragmentul de cod următor desparte în componente o dată calendaristică şi stochează rezultatul într-un vector:

```
$start_date = "2003-08-12";
$date_array = explode("-", $start_date);
// $date[0] == "2003"
// $date[1] == "08"
// $date[2] == "12"
```

Acum că aţi învăţat atâtea funcţii PHP dedicate şirurilor de caractere, este momentul să trecem la funcţiile care lucrează cu datele calendaristice şi cu orele.

Folosirea funcţiilor PHP dedicate datelor calendaristice şi orelor

Secţiunile care urmează prezintă funcţiile specifice limbajului PHP care operează cu date calendaristice şi ore. Încercaţi singur fiecare dintre listinguri pentru a vă da seama cât de simple şi puternice pot fi aceste funcţii.

Obţinerea datei curente cu ajutorul funcţiei `time()`

Funcţia `time()` a limbajului PHP vă oferă toate informaţiile de care aţi putea avea nevoie despre data şi ora curentă. Ea nu necesită nici un argument şi returnează un număr întreg. Pentru noi, oamenii, numărul returnat este destul de greu de suportat, dar el este, totuşi, foarte util.

```
echo time();
//exemplu de valoare returnata: 1060751270
//ceea ce inseamna 12 august 2003, ora 10:07 PM
```

Numărul întreg returnat de funcţia `time()` reprezintă numărul de secunde care au trecut de la miezul nopţii de 1 ianuarie 1970, conform orei Greenwich. Acest moment este considerat începutul erei Unix, iar numărului de secunde care s-au scurs de atunci poartă numele de *timestamp* (marcă temporală). PHP ne pune la dispoziţie instrumente excelente de conversie a unui *timestamp* în forme

inteligibile de către oameni. Chiar şi aşa, nu este folosirea acestui *timestamp* o modalitate mult prea complicată de stocare a unei date?

De fapt, adevărat este chiar contrariul. Pe baza unui singur număr, puteţi extrage o cantitate enormă de informaţii. Ba mai mult, datorită acestei mărci temporale, calculele cu date calendaristice sunt mult mai uşor de efectuat decât v-aţi închipui.

Gândiţi-vă la un sistem în care ar fi stocate zilele lunii, precum şi lunile şi anii. Acum imaginaţi-vă că trebuie să scrieţi un script care să adune o dată oarecare cu o zi. Dacă s-ar întâmpla ca data respectivă să fie 31 decembrie 1999, în loc să adunaţi data cu 1, ar trebui să scrieţi cod care să aducă ziua la valoarea 1, luna la valoarea ianuarie şi anul la valoarea 2000. Folosind un *timestamp*, nu trebuie decât să adăugaţi numărul de secunde corespunzător unei zile (60 × 60 × 24 sau 86 400) valorii curente şi aţi terminat. Puteţi converti, apoi, noua valoare în ceva mai inteligibil dacă doriţi.

Convertirea unui *timestamp* cu ajutorul funcţiei `getdate()`

Acum că aveţi un *timestamp* cu care să lucraţi, trebuie să îl convertiţi înainte de a îl prezenta utilizatorului. Funcţia `getdate()` acceptă drept argument opţional un *timestamp* şi returnează un vector asociativ care conţine informaţii despre dată. Dacă nu pasaţi *timestamp*-ul, `getdate()` converteşte *timestamp*-ul curent, aşa cum este el returnate de funcţia `time()`. Tabelul 8.3 enumeră elementele conţinute de vectorul returnat de `getdate()`.

TABELUL 8.3. Vectorul asociativ returnat de funcţia `getdate()`.

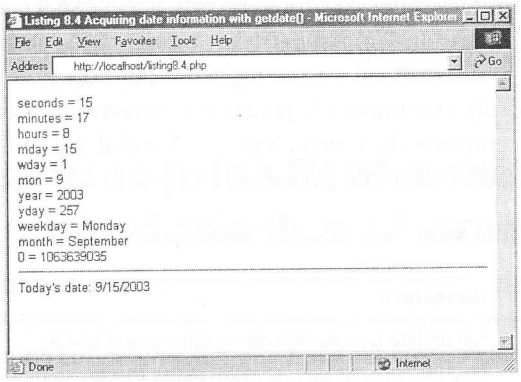
Cheie	Descriere	Exemplu
`seconds`	Secundele trecute faţă de minutul curent (0-59).	28
`minutes`	Minutele trecute faţă de ora curentă (0-59).	7
`hours`	Ora din zi (0-23).	12
`mday`	Ziua din lună (1-31).	20
`wday`	Ziua din săptămână (0-6).	4
`mon`	Luna din an (1-12).	1
`year`	Anul (4 cifre).	2000
`yday`	Ziua din an (0-365).	19
`weekday`	Ziua săptămânii (numele în engleză).	Thursday
`month`	Luna anului (numele în engleză).	January
`0`	*Timestamp*-ul.	948370048

În listingul 8.4 este folosită funcţia `getdate()` pe linia 7 şi se extrage informaţia prin intermediul unei instrucţiuni `foreach` pentru a afişa fiecare element (linia 8). Puteţi vedea un rezultat tipic în figura 8.5. Funcţia `getdate()` întoarce data în funcţie de fusul orar local.

LISTINGUL 8.4. Obținerea de informații temporale prin folosirea funcției `getdate()`.

```
1: <html>
2: <head>
3: <title>Listing 8.4 Acquiring date information with getdate()</title>
4: </head>
5: <body>
6: <?php
7: $date_array = getdate(); // no argument passed so today's date will be used
8: foreach ($date_array as $key => $val) {
9:     echo "$key = $val<br>";
10: }
11: ?>
12: <hr>
13: <?
14: echo "Today's date: ".$date_array['mon']."/".$date_array['mday']."/".
15:     $date_array['year']."<p>";
16: ?>
17: </body>
18: </html>
```

FIGURA 8.5. Folosirea funcției `getdate()`.



Convertirea unui *timestamp* folosind funcția `date()`

Funcția `getdate()` este folositoare atunci când trebuie să lucrați cu elementele pe care aceasta le returnează. Uneori, totuși, veți dori să afișați data sub forma unui șir de caractere. Funcția `date()` returnează un șir formatat care reprezintă data. Puteți exercita un grad enorm de control asupra formatului valorii returnate de funcția `date()` prin intermediul unui șir de caractere pe care trebuie să i-l pasați drept argument. Pe lângă acest șir de control al formatului, `date()` mai acceptă, opțional, un *timestamp*. Tabelul 8.4 enumeră câteva dintre codurile pe care șirul de control al formatului le poate conține. Puteți găsi lista lor completă la adresa `http://www.php.net/date`. Orice alte date pe care le includeți în șirul pasat funcției `date()` vor fi incluse în valoarea returnată.

TABELUL 8.4. Câteva coduri de format ce pot fi folosite cu funcția `date()`.

Cheie	Descriere	Exemplu
a	am sau pm (cu litere mici).	pm
A	AM sau PM (cu litere mari).	PM
d	Ziua lunii (număr cu zero nesemnificativ dacă are o singură cifră).	2Ø
D	Ziua săptămânii (trei litere, în engleză).	Thu
F	Numele lunii (în engleză).	January
h	Ora (formatul de 12 ore — cu zero nesemnificativ).	Ø8
H	Ora (formatul de 12 ore — cu zero nesemnificativ).	2Ø
g	Ora (formatul de 12 ore — fără zero nesemnificativ).	8
G	Ora (formatul de 24 de ore — fără zero nesemnificativ).	2Ø
i	Minutele.	47
j	Ziua lunii (fără zerouri nesemnificative).	2Ø
l	Ziua săptămânii (numele în engleză).	Thursday
L	An bisect (1 pentru da, 0 pentru nu).	1
m	Luna anului (număr — cu zero nesemnificativ).	Ø9
M	Luna anului (trei litere, în engleză).	Sep
n	Luna anului (număr — fără zerouri nesemnificative).	9
s	Secundele.	24
S	Sufixul ordinal (în engleză) corespunzător zilei.	th
r	Data complet standardizată, conform RFC 822 (`http://www.faqs.org/rfcs/rfc822.html`).	Mon, 15 Sep 2ØØ3 Ø8:25:55 –Ø7ØØ
U	*Timestamp*-ul.	1Ø63639555
y	Anul (două cifre).	ØØ
Y	Anul (patru cifre).	2ØØ3
z	Ziua anului (0-365).	257
Z	Diferența în secunde față de GMT.	Ø

Listingul 8.5 testează câteva dintre aceste coduri de format.

LISTINGUL 8.5. Formatarea unei date calendaristice cu ajutorul funcției `date()`.

```
1: <html>
2: <head>
3: <title>Listing 8.5 Formatting a date with date()</title>
4: </head>
5: <body>
6: <?php
```

LISTINGUL 8.5. (*continuare*)

```
 7: $time = time(); //stores the exact timestamp to use in this script
 8: echo date("m/d/y G.i:s", $time);
 9: echo "<br>";
10: echo "Today is ";
11: echo date("j of F Y, \a\\t g.i a", $time);
12: ?>
13: </body>
14: </html>
```

Listingul 8.5 apelează funcţia `date()` de două ori: o dată pe linia 8 pentru a afişa un data într-un format abreviat şi a doua oară pe linia 11, pentru un format mai lung. Salvaţi textul într-un fişier text denumit `listing8.5.php` şi deschideţi-l în browserul dumneavoastră. Data dumneavoastră va fi diferită, bineînţeles, dar iată un rezultat ilustrativ:

```
08/12/03 22.12:53
Today is 12 of August 2003, at 10.12 pm
```

Cu toate că are un aspect ermetic, şirul de control al formatului este uşor de construit. Dacă doriţi să adăugaţi un text care conţine litere ce fac parte dintre codurile de formatare, puteţi să evitaţi procesarea acestora plasând în faţa fiecăreia un caracter de eludare (\). Pentru caracterele care devin caractere de control numai atunci când sunt precedate de caracterul de eludare, trebuie să folosiţi două astfel de caractere. De exemplu, `\t` este caracterul de control care introduce un tabulator şi, pentru a vă asigura că va fi afişat tabulatorul, trebuie să folosiţi sintaxa `\\t`, ca în exemplul precedent.

Un alt exemplu ar fi situaţia în care adăugaţi şirului un cuvânt: de exemplu, `the`. Cuvântul `the` este compus din trei coduri de formatare, aşa că toate trebuie excluse de la procesare:

```
<?php
echo date("l \\t\h\e jS");
//afiseaza Monday the 15th
?>
```

De asemenea, reţineţi că funcţia `date()` returnează informaţii pe baza fusului orar local. Dacă doriţi să afişaţi o dată GMT, trebuie să folosiţi funcţia `gmtdate()`, care, în rest, funcţionează exact la fel.

Crearea de *timestamp*-uri cu ajutorul funcţiei `mktime()`

Aveţi deja informaţii despre ora curentă, dar nu puteţi să lucraţi, deocamdată, cu date arbitrare. Funcţia `mktime()` întoarce un *timestamp* pe care îl puteţi folosi, apoi, ca argument pentru funcţiile `date()` sau `getdate()`. Funcţia `mktime()` acceptă până la şase argumente numere întregi în următoarea ordine:

Ora

Minutul

Secunda

Luna

Ziua lunii

Anul

În listingul 8.6 este folosită funcţia `mktime()` pentru a se obţine un *timestamp* care, apoi, este pasat funcţiei `date()`.

LISTINGUL 8.6. Crearea unui *timestamp* cu ajutorul funcţiei `mktime()`.

```
 1: <html>
 2: <head>
 3: <title>Listing 8.6 Creating a timestamp with mktime()</title>
 4: </head>
 5: <body>
 6: <?php
 7: // creaza un timestamp pentru data de Aug 23 2003, ora 4.15 am
 8: $ts = mktime(4, 15, 0, 8, 23, 2003);
 9: echo date("m/d/y G.i:s", $ts);
10: echo "<br>";
11: echo "The date is ";
12: echo date("j of F Y, \a\\t g.i a", $ts );
13: ?>
14: </body>
15: </html>
```

Apelăm funcţia `mktime()` pe linia 8 şi atribuim valoarea pe care aceasta o returnează variabilei `$ts`. Apoi, folosim funcţia `date()` pe liniile 9 şi 12 pentru a afişa versiuni formatate ale datei utilizând variabila `$ts` ca argument. Puteţi alege să omiteţi o parte sau toate argumentele funcţiei `mktime()`, caz în care valoarea corespunzătoare datei şi orei curente va fi folosită. `mktime()` ajustează valorile care depăşesc intervalul acceptabil, astfel încât, dacă argumentul pentru oră este 25, el va fi tradus în 1:00 a.m. din ziua următoare celei specificate de argumentele pentru lună, zi şi an.

Salvaţi textul într-un fişier text denumit `listing8.5.php` şi deschideţi-l în browserul dumneavoastră. Ar trebui să vedeţi:

```
08/23/03 4.15:00
The date is 23 of August 2003, at 4.15 am
```

Testarea unei date cu ajutorul funcției `checkdate()`

S-ar putea să ajungeți în situația de a primi informații despre date calendaristice introduse de utilizator. Înainte de a lucra cu o dată astfel introdusă și înainte de a o stoca într-o bază de date, trebuie să verificați dacă data respectivă este validă. Funcția `checkdate()` acceptă trei argumente de tipul număr întreg: luna, ziua și anul. Funcția `checkdate()` returnează valoarea `true` dacă, în același timp, luna este între 1 și 12, ziua este acceptabilă pentru luna și anul date (ținând cont de anii bisecți), iar anul este între 0 și 32767. Fiți atenți, totuși: o dată calendaristică ar putea fi validă, dar, cu toate acestea, inacceptabilă pentru alte funcții care operează cu date. De exemplu, următoarele linii returnează valoarea `true`:

```
checkdate(4, 4, 1066)
```

Dacă ați încerca să folosiți funcția `mktime()` pentru a construi un *`timestamp`* folosind aceste valori, rezultatul pe care l-ați obține ar fi –1. O regulă simplă ar fi să nu utilizați `mktime()` cu ani de dinainte de 1902 și să fiți atenți atunci când folosiți funcții care operează cu date calendaristice pasându-le date de dinainte de 1970. Din moment ce epoca a început la 1 ianuarie 1970, orice dată anterioară acestei zile are un *`timestamp`* negativ, iar numerele negative nu reprezintă date valide.

Rezumat

În acest capitol am acoperit câteva dintre funcțiile care vă permit să controlați șirurile de caractere cu ajutorul scripturilor PHP. Ați învățat să formatați șirurile de caractere cu ajutorul funcțiilor `printf()` și `sprintf()`. Ar trebui să fiți capabili să folosiți aceste două funcții pentru a crea șiruri de caractere care transformă datele și le afișează. Ați învățat despre funcții care analizează șirurile de caractere. Ar trebui să puteți descoperi lungimea unui șir de caractere cu `strlen()`, să determinați prezența unui subșir cu `strpos()`, sau să extrageți un subșir cu `substr()`. Ar trebui, de asemenea, să puteți fragmenta un șir pe baza unor simboluri de delimitare cu ajutorul funcției `strtok()`.

Ați mai învățat și despre funcțiile care transformă șirurile de caractere. Puteți acum să eliminați spațiul alb de la începutul și de la sfârșitul unui șir cu `trim()`, `ltrim()` sau `rtrim()`. Puteți modifica capitalizarea literelor folosind `strtoupper()`, `strtolower()` sau `ucwords()`. Puteți înlocui toate aparițiile unui subșir în cadrul unui șir de caractere cu ajutorul funcției `str_replace()`.

Ați învățat, de asemenea, cum să folosiți diverse funcții PHP pentru a efectua operațiuni cu date calendaristice și ore. Funcția `time()` returnează un număr care semnifică data și ora curente, iar dumneavoastră puteți extrage dintr-un astfel de *`timestamp`* informații despre dată și oră folosind funcția `getdate()`, sau puteți folosi funcția `date()` ca să convertiți un *`timestamp`* într-un șir de caractere formatat. Ați învățat și cum să creați un *`timestamp`* cu funcția `mktime()` și cum să testați validitatea unei date calendaristice cu `checkdate()`. Veți învăța și alte funcții puternice dedicate datelor calendaristice în capitolul 15, intitulat *Comenzi SQL de bază*, și, foarte probabil, vă veți surprinde folosind MySQL în loc de PHP pentru a vă rezolva multe dintre nevoile legate de operarea cu date calendaristice.

Întrebări şi răspunsuri

Î. Mai există şi alte funcţii care mi-ar putea fi utile?

R. Da. PHP are mai bine de 60 de funcţii care lucrează cu şirurile de caractere! Puteţi citi totul despre ele în Manualul PHP on-line, la adresa `http://www.php.net/manual/ro/ref.strings.php`.

Atelier

Atelierul este conceput să anticipeze posibile întrebări, să vă ajute să recapitulaţi cele învăţate şi să începeţi să vă puneţi cunoştinţele în practică.

Test

1. Ce specificator de conversie aţi folosi în funcţia `printf()` pentru a afişa un număr întreg sub forma unui număr real în virgulă mobilă (`double`)? Indicaţi sintaxa completă care este necesară pentru a converti numărul întreg 33.
2. Cum aţi completa cu zerouri rezultatul conversiei pe care aţi efectuat-o la întrebarea 1 astfel încât partea de dinaintea virgulei să aibă patru caractere?
3. Cum aţi specifica o precizie de două zecimale pentru numărul real în virgulă mobilă pe care l-aţi formatat la întrebările anterioare?
4. Ce funcţie aţi folosi pentru a extrage un subşir dintr-un şir de caractere?
5. Cum aţi putea să eliminaţi spaţiul alb de la începutul unui şir de caractere?
6. Cum aţi împărţi un şir cu delimitatori transferând fragmentele într-un vector?
7. Folosind PHP, cum aţi prelua un *timestamp* Unix care să reprezinte data şi ora curente?
8. Care funcţie PHP acceptă drept argument un *timestamp* şi returnează un vector asociativ care reprezintă o anumită dată?
9. Care funcţie PHP se foloseşte pentru formatarea informaţiilor despre data calendaristică?
10. Ce funcţie PHP aţi folosi pentru a verifica validitatea unei date calendaristice?

Soluții

1. Specificatorul de conversie `f` este folosit pentru formatarea unui număr întreg sub formă de număr de tip `double`:

```
printf("%f", 33 );
```

2. Puteți completa rezultatul funcției `printf()` cu ajutorul specificatorului de completare—un spațiu sau un zero urmat de un număr reprezentând numărul de caractere la care vreți să ajungeți.

```
printf("%Ø4f", 33 );
```

3. Specificatorul de precizie constă dintr-un punct (.) urmat de un număr reprezentând precizia pe care doriți să o aplicați. Acestea ar trebui amplasate înainte de specificatorul de conversie:

```
printf("%Ø4.2f", 33 );
```

4. Funcția `substr()` extrage și returnează un subșir.
5. Funcția `ltrim()` elimină spațiul alb de la începutul unui șir.
6. Funcția `explode()` împarte un șir stocând fragmentele într-un vector.
7. Funcția `time()`.
8. Funcția `getdate()` returnează un vector asociativ ale cărui elemente conțin diverse aspecte ale datei respective.
9. Funcția `date()`.
10. Puteți verifica o dată calendaristică cu ajutorul funcției `checkdate()`.

Activități

1. Creați un formular de feedback care să accepte numele complet al utilizatorului și o adresă de e-mail. Folosiți funcțiile de conversie a capitalizării pentru a face ca prima literă a fiecărui nume pe care îl trimite utilizatorul să fie majusculă și afișați rezultatul înapoi în browser. Verificați dacă adresa de e-mail conține simbolul @ și afișați un avertisment în caz contrar.
2. Creați un vector de numere reale și întregi. Parcurgeți vectorul convertind fiecare element într-un număr în virgulă mobilă cu o precizie de 2 zecimale. Aliniați la dreapta rezultatul în interiorul unui câmp de 20 de caractere.
3. Creați un script care pentru anticiparea zilei de naștere. Utilizatorul introduce luna, ziua și anul, iar mesajul afișat îi spune utilizatorului câte zile, ore, minute și secunde mai sunt până atunci.

CAPITOLUL 9
Folosirea formularelor

Până acum, exemplelor de cod PHP din această carte le-a lipsit o dimensiune esențială. Sigur, știți chestiunile de bază, puteți să definiți variabile și vectori, să creați și să apelați funcții și să lucrați cu șirurile de caractere. Dar toate astea sunt lipsite de sens dacă utilizatorii nu pot să comunice cu programul dumneavoastră și să îi ofere acestuia informații. În acest capitol, vom analiza strategii de preluare și procesare a datelor introduse de utilizator. Pe Web, formularele HTML reprezintă mijlocul cel mai important prin care volume importante de informații ajung de la utilizator la server.

În acest capitol, veți învăța cum să:

- accesați informația din câmpurile formularelor;
- lucrați cu elementele de formular care permit selecții multiple;
- creați un singur document care să conțină atât formularul HTML cât și codul PHP care preia datele trimise;
- păstrați starea prin intermediul câmpurilor ascunse;
- redirecționați utilizaforul către o pagină nouă;
- construiți formulare HTML și cod PHP pentru trimiterea de mesaje e-mail;
- construiți formulare HTML care să încarce fișiere și cum să scrieți cod PHP care să opereze cu acestea.

Crearea unui formular simplu de introducere a datelor

Deocamdată, hai să ținem codul HTML separat de cel PHP. Listingul 9.1 reprezintă un formular HTML simplu.

LISTINGUL 9.1. Un formular HTML simplu.

```
1: <html>
2: <head>
3: <title>Listing 9.1 A simple HTML form</title>
4: </head>
5: <body>
6: <form action="listing9.2.php" method="POST">
```

LISTINGUL 9.1. (*continuare*)

```
 7: <p><strong>Name:</strong><br>
 8: <input type="text" name="user"></p>
 9: <p><strong>Address:</strong><br>
10: <textarea name="address" rows="5" cols="40"></textarea></p>
11: <p><input type="submit" value="send"></p>
12: </form>
13: </body>
14: </html>
```

Introduceţi aceste linii într-un fişier text denumit `listing9.1.html` şi amplasaţi fişierul în directorul rădăcină a serverului dumneavoastră Web. Listingul defineşte un formular care conţine un câmp de tip text denumit `"user"` — linia 8 — o zonă de introducere a textului denumită `"address"`, pe linia 10 şi un buton de trimitere a datelor pe linia 12. Argumentul `ACTION` al elementului `FORM` indică un fişier cu numele `listing9.2.php`, care prelucrează informaţia din formular. Metoda acestui formular este `POST`, aşa că variabilele sunt stocate în vectorul superglobal `$_POST`.

În listingul 9.2 se găseşte codul care primeşte datele introduse de utilizator.

LISTINGUL 9.2. Citirea datelor introduse într-un formular.

```
 1: <html>
 2: <head>
 3: <title>Listing 9.2 Reading input from a form </title>
 4: </head>
 5: <body>
 6: <?php
 7: echo "<p>Welcome <b>$_POST[user]</b></p>";
 8: echo "<p>Your address is:<br><b>$_POST[address]</b></p>";
 9: ?>
10: </body>
11: </html>
```

Plasaţi aceste linii într-un fişier text denumit `listing9.2.php` şi amplasaţi fişierul în directorul rădăcină a serverului dumneavoastră Web. Acum, accesaţi formularul (fişierul `listing9.1.html`) şi veţi vedea ceva similar cu figura 9.1.

Scriptul din listingul 9.2 este primul script din această carte care nu a fost conceput pentru a fi lansat dând clic pe un link sau tastând numele direct în bara de adrese a browserului, ci pentru a fi apelat atunci când utilizatorul trimite formularul creat în listingul 9.1.

În interiorul codului, accesăm două variabile: `$_POST[user]` şi `$_POST[address]`. Acestea sunt referinţe la vectorul superglobal `$_POST`, care conţine valorile pe care utilizator le-a introdus în câmpul de text `user` şi în zona de introducere a textului `address`. Folosirea formularelor este, după cum puteţi observa, cât se poate de simplă în PHP.

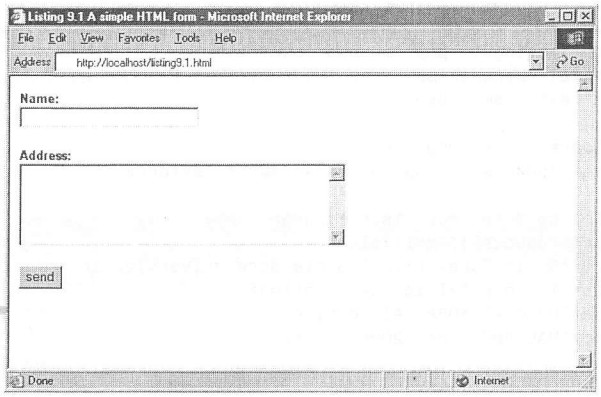


FIGURA 9.1. Formularul creat în listingul 9.1.

Introduceţi nişte informaţii în câmpurile formularului şi daţi clic pe butonul send (trimite). Ar trebui să vedeţi afişate datele pe care le-aţi introdus.

Apropo

Aţi putea folosi şi metoda GET pentru acest formular (şi pentru altele). POST poate opera cu mai multe date decât GET şi nu pasează datele prin intermediul şirului de interogare. Dacă folosiţi metoda GET, totuşi, aveţi grijă să schimbaţi referinţa către vectorul superglobal $_POST în $_GET.

Accesarea datelor introduse în formular folosind vectori definiţi de utilizator

Exemplul precedent v-a arătat cum să culegeţi informaţii de la elementele HTML care trimit o singură valoare pentru fiecare nume de element. Acest lucru nu rezolvă problema elementelor de tip SELECT, în cazul cărora utilizatorul poate alege una sau mai multe valori. Dacă denumim elementul SELECT cu un nume normal, cum ar fi:

```
<select name="products" multiple>
```

scriptul care primeşte datele are acces numai la una dintre valorile corespunzătoare acestui nume. Putem modifica acest comportament redenumind un element de acest fel astfel încât numele său să se termine cu o pereche de paranteze pătrate. Am făcut acest lucru în listingul 9.3.

LISTINGUL 9.3. Un formular HTML care include un element de tip SELECT.

```
1: <html>
2: <head>
3: <title>Listing 9.3 An HTML form including a SELECT element</title>
4: </head>
5: <body>
```

LISTINGUL 9.3. (*continuare*)

```
 6: <form action="listing9.4.php" method="POST">
 7: <p><strong>Name:</strong><br>
 8: <input type="text" name="user">
 9:
10: <p><strong>Address:</strong><br>
11: <textarea name="address" rows="5" cols="40"></textarea>
12:
13: <p><strong>Select Some Products:</strong> <br>
14: <select name="products[]" multiple>
15: <option value="Sonic Screwdriver">Sonic Screwdriver</option>
16: <option value="Tricoder">Tricorder</option>
17: <option value="ORAC AI">ORAC AI</option>
18: <option value="HAL 2000">HAL 2000</option>
19: </select>
20:
21: <p><input type="submit" value="send"></p>
22: </form>
23: </body>
24: </html>
```

Introduceţi aceste linii într-un fişier text denumit `listing9.3.html` şi amplasaţi fişierul în directorul rădăcină a serverului dumneavoastră Web. Apoi, în scriptul care procesează datele transmise de către formular, descoperim că acele date introduse de utilizator în elementul `"products[]"` al formularului creat pe linia 14 sunt disponibile într-un vector denumit `$_POST[products]`. Din cauză că `products[]` este un element de tip `SELECT`, îi oferim utilizatorului posibilitatea de a selecta mai multe variante cu ajutorul elementelor `OPTION` de pe liniile 15–18. Demonstrăm faptul că opţiunile utilizatorului sunt disponibile într-un vector în listingul 9.4.

LISTINGUL 9.4. Citirea datelor introduse de utilizator în formularul din listingul 9.3.

```
 1: <html>
 2: <head>
 3: <title>Listing 9.4 Reading input from the form in Listing 9.3</title>
 4: </head>
 5: <body>
 6: <?php
 7: echo "<p>Welcome <b>$_POST[user]</b></p>";
 8:  echo "<p>Your address is:<br><b>$_POST[address]</b></p>";
 9:  echo "<p>Your product choices are:<br>";
10: if (!empty($_POST[products])) {
11:     echo "<ul>";
12:     foreach ($_POST[products] as $value) {
13:        echo "<li>$value";
14:     }
15:     echo "</ul>";
16: }
17: ?>
18: </body>
19: </html>
```

Puneţi aceste linii într-un fişier text denumit `listing9.4.php` şi amplasaţi fişierul în directorul rădăcină a serverului dumneavoastră Web. Acum accesaţi formularul din listingul 9.3 cu browserul dumneavoastră. Figura 9.2 ilustrează acest formular.

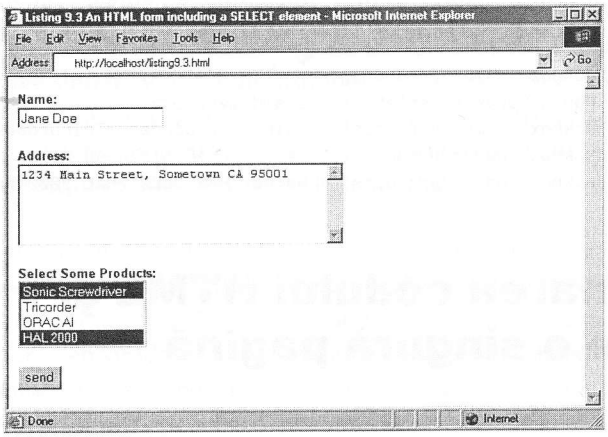


FIGURA 9.2. Formularul creat în listingul 9.3.

Pe linia 7 a scriptului din listingul 9.4, accesăm variabila `$_POST[user]`, care derivă din elementul user al formularului. Pe linia 10, testăm existenţa variabilei `$_POST[products]`. Dacă `$_POST[products]` este prezentă, o parcurgem element cu element pe linia 12 şi afişăm fiecare alegere în browser pe linia 13. Textul stocat în atributul value al elementului `OPTION` selectat devine una dintre valorile stocate în vector.

Transmiteţi formularul şi veţi vedea ceva asemănător cu imaginea din figura 9.3.

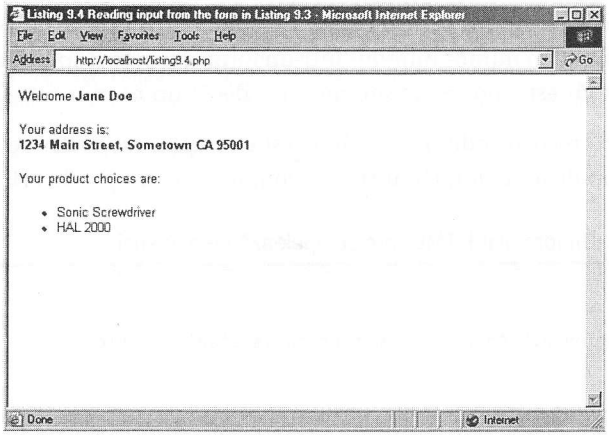


FIGURA 9.3. Exemplu de rezultat al listingului 9.4.

Deşi tehnica de parcurgere iterativă este folositoare în special pentru elementul SELECT, aceasta funcţionează şi în cazul altor tipuri de elemente ale formularelor. De exemplu, dacă creăm mai multe casete de validare cu acelaşi nume, putem să îi permitem utilizatorului să aleagă mai multe valori în cadrul aceluiaşi nume de câmp. Atâta vreme cât numele pe care îl alegeţi se termină cu o pereche de paranteze pătrate, PHP concatenează datele introduse de utilizator pentru acest câmp sub forma unui vector. Putem să înlocuim elementele SELECT de pe liniile 15–18 din listingul 9.3 cu o serie de casete de validare, efectul rămânând acelaşi:

```
<input type="checkbox" name="products[]"
       value="Sonic Screwdriver">Sonic Screwdriver<br>
<input type="checkbox" name="products[]" value="Tricorder">Tricorder<br>
<input type="checkbox" name="products[]" value="ORAC AI">ORAC AI<br>
<input type="checkbox" name="products[]" value="HAL 2000">HAL 2000<br>
```

Combinarea codului HTML şi PHP pe o singură pagină

În anumite circumstanţe, s-ar putea să doriţi să includeţi cod care să prelucreze un formular pe aceeaşi pagină cu codul HTML static. O asemenea combinaţie se poate dovedi utilă atunci când trebuie să îi prezentaţi utilizatorului acelaşi formular de mai multe ori. Dacă aţi scrie întreaga pagină dinamic, aţi avea mai multă flexibilitate, evident, dar aţi pierde ocazia de a folosi una dintre cele mai puternice caracteristici ale limbajului PHP. Cu cât lăsaţi mai mult cod HTML standard în paginile dumneavoastră, cu atât le este mai uşor designerilor şi celor care construiesc paginile să le modifice fără să aibă nevoie de dumneavoastră. Ar trebui, totuşi, să evitaţi să împrăştiaţi fragmente substanţiale de PHP printre documentele dumneavoastră. Dacă faceţi acest lucru, codul va fi greu de citit şi întreţinut. Ori de câte ori este posibil, trebuie să creaţi funcţii care să poată fi apelate de către codul HTML şi care să poată fi reutilizate la alte proiecte.

Pentru exemplele următoare, imaginaţi-vă că lucrăm la un site educaţional de matematică elementară pentru copiii preşcolari şi că ni s-a cerut să creăm un script care să preia un număr introdus într-un formular şi să îi spună utilizatorului dacă acest număr este mai mare sau mai mic decât un număr întreg predefinit.

Listingul 9.5 creează codul HTML. În acest exemplu, nu avem nevoie decât de un singur câmp de text, dar, chiar şi aşa, vom include un pic de PHP.

LISTINGUL 9.5. Un formular HTML care se apelează pe el însuşi.

```
1: <html>
2: <head>
3: <title>Listing 9.5 An HTML form that calls itself</title>
4: </head>
5: <body>
6: <form action="<?php echo $_SERVER[PHP_SELF] ?>" method="POST">
```

LISTINGUL 9.5. (*continuare*)

```
 7: <p><strong>Type your guess here:</strong> <input type="text" name="guess"></p>
 8: <p><input type="submit" value="submit your guess"></p>
 9: </form>
10: </body>
11: </html>
```

Acţiunea acestui script este `$_SERVER[PHP_SELF]`. Această variabilă este echivalentă cu numele scriptului curent. Cu alte cuvinte, acţiunea îi spune scriptului să se re-încarce singur. Scriptul din listingul 9.5 nu afişează nici un rezultat. Cu toate acestea, dacă puneţi scriptul în serverul dumneavoastră Web, accesaţi pagina şi vizualizaţi codul sursă al acesteia, veţi observa că parametrul `action` al formularului conţine acum chiar numele scriptului.

În listingul 9.6 am început prin construirea unui element PHP în pagină. Mai întâi, trebuie să definim numărul pe care utilizatorul trebuie să îl ghicească. Într-o versiune complet funcţională, probabil că am genera numărul în mod aleatoriu, dar, deocamdată, vom simplifica lucrurile. Atribuim valoarea 42 variabilei `$num_to_guess` pe linia 2. După aceasta, trebuie să stabilim dacă formularul a fost sau nu transmis; altminteri, am încerca să evaluăm variabile care nu sunt încă disponibile. Putem verifica dacă formularul a fost trimis testând existenţa variabilei `$_POST[guess]`, care devine disponibilă abia după ce scriptul a primit un parametru `guess`. Dacă `$_POST[guess]` nu este prezent, atunci putem să presupunem că utilizatorul abia a ajuns pe pagină fără să mai fi transmis vreun formular. Dacă variabila este prezentă, putem să testăm valoare pe care o conţine. Verificarea prezenţei variabilei `$_POST[guess]` are loc pe linia 4.

LISTINGUL 9.6. Un script PHP de ghicire a unui număr.

```
 1: <?php
 2: $num_to_guess = 42;
 3: $message = "";
 4: if (!isset($_POST[guess])) {
 5:    $message = "Welcome to the guessing machine!";
 6: } else if ($_POST[guess] > $num_to_guess) {
 7:    $message = "$_POST[guess] is too big! Try a smaller number";
 8: } else if ($_POST[guess] < $num_to_guess) {
 9:    $message = "$_POST[guess] is too small! Try a larger number";
10: } else { // must be equivalent
11:    $message = "Well done!";
12: }
13: ?>
14: <html>
15: <head>
16: <title>Listing 9.6 A PHP number guessing script</title>
17: </head>
18: <body>
19: <h1>
20: <?php echo $message ?>
21: </h1>
```

LISTINGUL 9.6. *(continuare)*

```
22: <form action="<?php echo $_SERVER[PHP_SELF] ?>" method="POST">
23: <p><strong>Type your guess here:</strong> <input type="text" name="guess"></
p>
24: <p><input type="submit" value="submit your guess"></p>
25: </form>
26: </body>
27: </html>
```

Puneţi aceste linii într-un fişier text denumit `listing9.6.php` şi amplasaţi fişierul în directorul rădăcină a serverului dumneavoastră Web. Acum accesaţi scriptul în browserul dumneavoastră şi ar trebui să vedeţi ceva asemănător cu imaginea din figura 9.4.

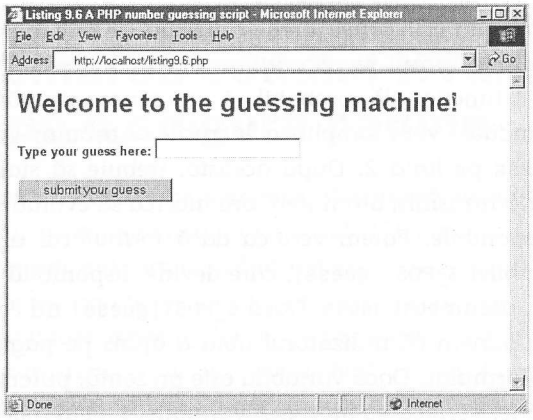


FIGURA 9.4. Formularul creat în listingul 9.6.

Cea mai mare parte a scriptului constă dintr-o instrucţiune condiţională `if` care determină ce şir de caractere trebuie atribuit variabilei `$message`. Dacă variabila `$_POST[guess]` nu a fost iniţializată, presupunem că utilizatorul respectiv a sosit pentru prima oară pe pagină şi atribuim ca valoare un text de bun venit variabilei `$message` (linia 5).

În caz contrar, testăm dacă variabila `$_POST[guess]` are aceeaşi valoare ca variabila `$num_to_guess` şi atribuim variabilei `$message` sfatul corespunzător. Verificăm dacă `$_POST[guess]` este mai mare decât `$num_to_guess` pe linia 6 şi dacă este mai mic decât `$num_to_guess` pe linia 8. Dacă variabila `$_POST[guess]` nu este nici mai mare şi nici mai mică decât `$num_to_guess`, putem presupune că cele două sunt egale şi atribuim un mesaj de felicitare variabilei `$message` (linia 11). Tot ce mai avem de făcut acum este să afişăm variabila `$message` în cadrul corpului documentului HTML.

Mai sunt câteva adăugiri pe care le-aţi putea face, dar, probabil, vă daţi seama de pe-acum ce uşor este să predaţi această pagină unui designer, care poate să o înfrumuseţeze fără să deranjeze programarea în vreun fel.

Folosirea câmpurilor ascunse pentru păstrarea stării

Scriptul din listingul 9.6 nu are cum să ştie cât de multe încercări a făcut un utilizator, dar putem folosi un câmp ascuns pentru a le număra. Un câmp ascuns se comportă exact ca un câmp de text, doar că utilizatorul nu îl poate vedea decât dacă vizualizează sursa HTML a documentului care îl conţine. Listingul 9.7 adaugă un câmp ascuns scriptului de ghicire a unui număr, precum şi ceva cod PHP care să lucreze cu acesta.

LISTINGUL 9.7. Păstrarea stării prin utilizarea unui câmp ascuns.

```
 1: <?php
 2: $num_to_guess = 42;
 3: $num_tries = (isset($_POST[num_tries])) ? $num_tries  + 1 : 0;
 4: $message = "";
 5: if (!isset($_POST[guess])) {
 6:    $message = "Welcome to the guessing machine!";
 7: } else if ($_POST[guess] > $num_to_guess) {
 8:    $message = "$_POST[guess] is too big! Try a smaller number";
 9: } else if ($_POST[guess] < $num_to_guess) {
10:    $message = "$_POST[guess] is too small! Try a larger number";
11: } else { // must be equivalent
12:    $message = "Well done!";
13: }
14: $guess = $_POST[guess];
15: ?>
16: <html>
17: <head>
18: <title>Listing 9.7 Saving state with a hidden field</title>
19: </head>
20: <body>
21: <h1>
22: <?php echo $message ?>
23: </h1>
24: <p><strong>Guess number:</strong> <?php echo $num_tries ?></p>
25: <form action="<?php echo $_SERVER[PHP_SELF] ?>" method="POST">
26: <p><strong>Type your guess here:</strong>
27: <input type="text" name="guess" value="<?php echo $guess ?>">
28: <input type="hidden" name="num_tries" value="<?php echo $num_tries ?>">
29: <p><input type="submit" value="submit your guess"></p>
30: </form>
31: </body>
32: </html>
```

Câmpul ascuns de pe linia 28 primeşte denumirea `"num_tries"`. Folosim, de asemenea, cod PHP pentru a scrie această valoare. Şi dacă tot suntem aici, putem face acelaşi lucru şi cu câmpul de pe linia 27 astfel încât utilizatorul să poată vedea oricând ultimul număr pe care l-a încercat. Această tehnică este utilă pentru scripturile care prelucrează date de intrare. Dacă respingem, dintr-un motiv sau altul, datele transmise de utilizator, putem, cel puţin, să îi permitem să le modifice pe ultimele introduse.

În cadrul codului PHP principal, folosim un operator ternar pentru a incrementa variabila `$num_tries`. Dacă variabila `$num_tries` este iniţializată, îi adăugăm valoarea unu şi îi re-atribuim valoarea incrementată; altminteri, iniţializăm variabila `$num_tries` la valoarea 0. În corpul documentului HTML, putem acum să îi raportăm utilizatorului câte încercări a făcut.

Puneţi aceste linii într-un fişier text denumit `listing9.7.php` şi amplasaţi fişierul în directorul rădăcină a serverului dumneavoastră Web. Accesaţi formularul de câteva ori şi încercaţi să ghiciţi numărul (prefaceţi-vă că nu îl ştiaţi dinainte).

Redirecţionarea utilizatorului

Scriptul nostru simplu are, încă, un dezavantaj major. Formularul este reîncărcat indiferent dacă utilizator ghiceşte sau nu corect. Din cauza codului HTML static, ne este greu să evităm să scriem din nou întreaga pagină. Putem, totuşi, să redirecţionăm utilizatorul către o pagină de felicitare, evitând această problemă cu totul.

Atunci când un script de pe server comunică cu un client, el trebuie mai întâi să trimită nişte anteturi care furnizează informaţii despre tipul de document care urmează. PHP se ocupă, de obicei, automat de acest lucru, dar puteţi de asemenea să trimiteţi anteturile dumneavoastră proprii cu ajutorul funcţiei `header()`.

Pentru a apela funcţia `header()`, trebuie să vă asiguraţi că nu aţi trimis nici un fel de date către browser până în acel moment. În momentul în care primul fragment de conţinut este trimis către browser, PHP îşi trimite anteturile şi este deja prea târziu ca să le mai trimiteţi pe ale dumneavoastră. Orice porţiune a documentului, chiar şi un singur marcaj de sfârşit de linie sau un spaţiu care nu este cuprins între marcajele scriptului dumneavoastră face ca anteturile să fie trimise. Dacă intenţionaţi să folosiţi funcţia `header()` într-un script, trebuie să fiţi absolut sigur că nimic nu se găseşte înaintea codului PHP care conţine apelul către această funcţie. Ar mai trebui, de asemenea, să verificaţi diversele biblioteci incluse pe care le-aţi putea, eventual, utiliza.

Listingul 9.8 arată anteturile standard care sunt trimise de obicei de PHP, începând cu linia 3, drept răspuns la solicitarea din linia 1.

LISTINGUL 9.8. Anteturile de server tipice pe care le trimite un script PHP.

```
1: HTTP/1.1 200 OK
2: Date: Mon,  11 Aug 2003 12:32:28 GMT
3: Server: Apache/2.0.44 (Unix) mod_ssl/2.0.44 OpenSSL/0.9.7a PHP/4.3.2
4: X-Powered-By: PHP/4.3.2
5: Connection: close
6: Content-Type: text/html
```

Trimiţând un antet `Location` în locul celui trimis implicit de PHP, puteţi redirecţiona browserul către o pagină nouă, ca de pildă:

```
header("Location: http://www.samspublishing.com");
```

Presupunând că am creat deja o pagină frumoasă denumită `congrats.html`, putem să modificăm scriptul nostru de ghicire a unui număr astfel încât utilizatorul să fie redirecţionat dacă ghiceşte corect, aşa cum se vede în listingul 9.9.

LISTINGUL 9.9. Folosirea funcţiei `header()` pentru a trimite anteturi brute.

```
 1: <?php
 2: $num_to_guess = 42;
 3: $num_tries = (isset($_POST[num_tries])) ? $num_tries + 1: 0;
 4: $message = "";
 5: if (!isset($_POST[guess])) {
 6:    $message = "Welcome to the guessing machine!";
 7: } elseif ($_POST[guess] > $num_to_guess) {
 8:    $message = "$_POST[guess] is too big! Try a smaller number";
 9: } elseif ($_POST[guess] < $num_to_guess) {
10:    $message = "$_POST[guess] is too small! Try a larger number";
11: } else { // must be equivalent
12:    header("Location: congrats.html");
13:    exit;
14: }
15: $guess = $_POST[guess];
16: ?>
17: <html>
18: <head>
19: <title>Listing 9.9 Using header() to Send Raw Headers</title>
20: </head>
21: <body>
22: <h1>
23: <?php echo $message ?>
24: </h1>
25: <p><strong>Guess number:</strong> <?php echo $num_tries ?></p>
26: <form action="<?php echo $_SERVER[PHP_SELF] ?>" method="POST">
27: <p><strong>Type your guess here:</strong>
28: <input type="text" name="guess" value="<?php echo $guess ?>">
29: <input type="hidden" name="num_tries" value="<?php echo $num_tries ?>">
30: <p><input type="submit" value="submit your guess"></p>
31: </form>
32: </body>
33: </html>
```

Clauza `else` a instrucţiunii noastre condiţionale `if` de pe linia 11 face, acum, ca browserul să solicite serverului pagina `congrats.html`. Ne asigurăm că orice transmisiune de date dinspre pagina curentă este abandonată folosind instrucţiunea `exit` pe linia 13, care opreşte imediat execuţia şi transmiterea de date, indiferent că este vorba de PHP sau HTML.

Trimiterea unui mesaj e-mail la primirea formularului

Am văzut deja cum să preluăm răspunsurile dintr-un formular şi să le afişăm pe ecran. Sunteţi la un pas de a putea să trimiteţi aceste răspunsuri prin e-mail, după cum veţi vedea imediat. Înainte să învăţaţi cum să trimiteţi mesaje e-mail, totuşi, citiţi secţiunea următoare pentru a vă asigura că sistemul este configurat corespunzător.

Configurarea sistemului pentru funcția `mail()`

Înainte de a putea folosi funcţia `mail()` pentru a trimite mesaje e-mail, trebuie să existe câteva directive în fişierul `php.ini` astfel încât funcţia să opereze corespunzător. Deschideţi `php.ini` cu un editor de text şi căutaţi liniile următoare:

```
[mail function]
; For Win32 only.
SMTP = localhost

; For Win32 only.
sendmail_from = me@localhost.com

; For Unix only.  You may supply arguments as well (default: "sendmail -t -i").
;sendmail_path =
```

Dacă folosiţi Windows drept platformă pentru serverul Web, primele două directive vi se aplică dumneavoastră. Pentru ca funcţia `mail()` să poată trimite mesaje, ea trebuie să poată accesa un server de e-mail valid. Dacă plănuiţi să folosiţi serverul de trimitere a mesajelor e-mail al ISP-ului dumneavoastră (în exemplul de mai jos utilizăm EarthLink), intrarea din `php.ini` trebuie să arate astfel:

```
SMTP = mail.earthlink.net
```

Cea de-a doua directivă de configurare este `sendmail_from`, care reprezintă adresa de e-mail folosită în antetul From (de la) al mesajelor trimise. Aceasta poate fi schimbată chiar în script, dar joacă, de obicei, rolul de adresă implicită. De exemplu:

```
sendmail_from = youraddress@yourdomain.com
```

O regulă de bun simţ pentru cei ce folosesc sistemul de operare Windows este să folosiţi pentru valoarea directivei SMTP din `php.ini` acelaşi server de trimitere a mesajelor e-mail cu care este configurat clientul dumneavoastră de e-mail.

Dacă serverul dumneavoastră Web rulează pe o platformă Unix/Linux, veţi folosi funcţionalitatea de trimitere a mesajelor e-mail a maşinii respective. În acest caz, vi se aplică numai ultima directivă: `sendmail_path`. Valoarea implicită este `sendmail -t -i`, dar, dacă `sendmail` se găseşte într-un alt loc sau dacă trebuie să specificaţi alte argumente, faceţi acest lucru cu încredere, aşa cum puteţi observa în exemplul de mai jos, care nu foloseşte valori reale:

```
sendmail_path = /opt/sendmail -odd –arguments
```

După ce aţi făcut schimbări în fişierul `php.ini`, indiferent de platformă, trebuie să reporniţi procesul serverului Web pentru ca schimbările să intre în vigoare.

Crearea formularului

În listingul 9.10 puteţi vedea codul HTML pentru crearea unui formular de feedback simplu. Acest formular are drept acţiune fişierul `listing9.11.php`, pe care îl vom crea în secţiunea următoare. Câmpurile sunt foarte simple: linia 7 conţine un câmp pentru nume, linia 8 conţine adresa de e-mail, iar linia 10 conţine zona de introducere a textului pentru mesajul utilizatorului.

LISTINGUL 9.10. Crearea unui formular de feedback simplu.

```
 1: <HTML>
 2: <HEAD>
 3: <TITLE>E-Mail Form</TITLE>
 4: </HEAD>
 5: <BODY>
 6: <FORM action="listing9.11.php" method="POST">
 7: <p><strong>Your Name:</strong><br> <INPUT type="text" size="25"
    name="name"></p>
 8: <p><strong>Your E-Mail Address:</strong><br> <INPUT type="text"
    size="25" name="email"></p>
 9: <p><strong>Message:</strong><br>
10: <textarea name="message" cols=30 rows=5></textarea></p>
11: <p><INPUT type="submit" value="send"></p>
12: </FORM>
13: </BODY>
14: </HTML>
```

Introduceţi aceste linii într-un fişier text denumit `listing9.10.html` şi amplasaţi fişierul în directorul rădăcină a serverului dumneavoastră Web. Acum accesaţi scriptul cu browserul dumneavoastră şi ar trebui să vedeţi ceva similar imaginii din figura 9.5.

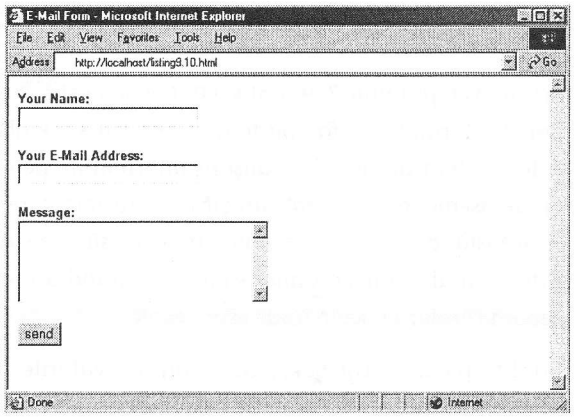


FIGURA 9.5. Formularul corespunzător codului din listingul 9.10.

În secţiunea următoare veţi crea un script care trimite acest formular către un destinatar.

Crearea scriptului care să trimită mesajul e-mail

Scriptul din listingul 9.11 este doar foarte puţin diferit din punct de vedere conceptual de scriptul din listingul 9.4, care nu făcea decât să afişeze răspunsurile din formular pe ecran. În acest nou script, pe lângă afişarea răspunsurilor pe ecran, acestea sunt şi trimise către o adresă de e-mail.

LISTINGUL 9.11. Trimiterea formularului de feedback prin e-mail.

```
 1: <html>
 2: <head>
 3: <title>Listing 9.11 Sending mail from the form in Listing 9.10</title>
 4: </head>
 5: <body>
 6: <?php
 7: echo "<p>Thank you, <b>$_POST[name]</b>, for your message!</p>";
 8: echo "<p>Your e-mail address is: <b>$_POST[email]</b></p>";
 9: echo "<p>Your message was:<br>";
10: echo "$_POST[message] </p>";
11: //incepe construirea antetului mesajului
12: $msg = "Name:      $_POST[name]\n";
13: $msg .= "E-Mail:  $_POST[email]\n";
14: $msg .= "Message: $_POST[message]\n";
15: //se configureaza mesajul
16: $recipient = "you@yourdomain.com";
17: $subject = "Form Submission Results";
18: $mailheaders = "From: My Web Site <defaultaddress@yourdomain.com> \n";
19: $mailheaders .= "Reply-To: $_POST[email]";
20: //mesajul este trimis
21: mail($recipient, $subject, $msg, $mailheaders);
22: ?>
23: </body>
24: </html>
```

Variabilele pe care le folosiţi pe liniile 7–9 sunt `$_POST[email]` şi `$_POST[message]` — numele câmpurilor din formular şi fac parte din vectorul superglobal `$_POST`. Acesta este un lucru foarte bun atunci când afişăm informaţiile pe ecran, dar în acest script va trebui, de asemenea, să creaţi un şir de caractere care să fie trimis prin e-mail. Pentru această operaţiune, trebuie, practic, să construiţi mesajul concatenând şirurile din formular într-un singur şir lung, folosind caracterul de linie nouă (`\n`) pentru a separa rândurile acolo unde este nevoie.

Liniile de la 12 la 14 creează şirul `$msg`, care conţine valorile introduse de utilizator în câmpurile formularului. Acest şir de caractere este cel care este trimis prin e-mail. Observaţi folosirea operatorului de concatenare (`.=`) atunci când se adaugă şiruri la variabila `$msg`, pe liniile 13 şi 14.

Liniile 16 şi 17 reprezintă variabile statice care conţin destinatarul şi subiectul mesajelor e-mail. Fireşte, trebuie să înlocuiţi şirul `you@yourdomain.com` cu adresa dumneavoastră reală de e-mail. Dacă vreţi să schimbaţi subiectul, faceţi-o fără grijă!

Liniile 18 şi 19 iniţializează câteva anteturi ale mesajului e-mail, şi anume From: (de la), Reply-to: (răspunde la). Aţi putea pune orice valoare în antetul From:, aceasta este informaţia care este afişată în coloana From sau Sender (de la) a aplicaţiei dumneavoastră de e-mail atunci când primiţi mesajele.

Apropo

Dacă serverul dumneavoastră de trimitere a mesajelor e-mail este o maşină Windows, caracterul de linie nouă, `\n`, trebuie înlocuit cu secvenţa `\r\n`.

Funcţia `mail()` acceptă patru argumente: destinatarul, subiectul, mesajul şi alte anteturi adiţionale. Ordinea acestor parametri poate fi văzută pe linia 21, iar scriptul dumneavoastră este complet după ce închideţi blocul de cod PHP şi elementele HTML pe liniile 22–24.

Puneţi aceste linii într-un fişier text denumit `listing9.11.html` şi amplasaţi fişierul în directorul rădăcină a serverului dumneavoastră Web. Folosiţi browserul pentru a încărca din nou formularul, introduceţi nişte informaţii şi daţi clic pe butonul de trimitere. Ar trebui să vedeţi în browserul dumneavoastră ceva similar imaginii din figura 9.6.

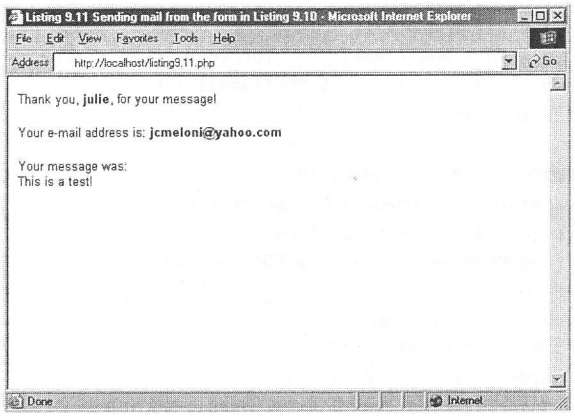


FIGURA 9.6. Un exemplu de rezultat al listingului 9.11.

Dacă vă verificaţi, apoi, căsuţa de e-mail, acolo ar trebui să fie un mesaj nou. Acesta ar trebui să arate ca în figura 9.7.

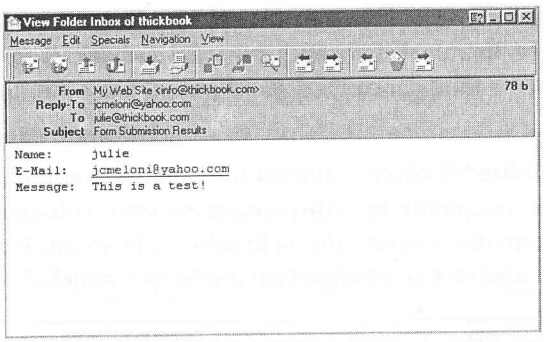


FIGURA 9.7. Mesajul e-mail trimis de codul din listingul 9.11

Formatarea mesajului e-mail cu HTML

„Şmecheria" prin care puteţi trimite mesaje e-mail formatate în HTML nu este deloc o şmecherie. De fapt, nu este nevoie decât să scrieţi HTML-ul efectiv şi să modificaţi anteturile trimise de funcţia `mail()`. În listingul 9.12, o variaţiune a listingului 9.11, puteţi vedea schimbările făcute pe liniile 12–14 şi 18–19.

LISTINGUL 9.12. Trimiterea formularului de feedback — versiunea HTML

```
 1: <html>
 2: <head>
 3: <title>Listing 9.12 Sending the Simple Feedback Form – HTML Version</title>
 4: </head>
 5: <body>
 6: <?php
 7: echo "<p>Thank you, <b>$_POST[name]</b>, for your message!</p>";
 8: echo "<p>Your e-mail address is: <b>$_POST[email]</b></p>";
 9: echo "<p>Your message was:<br>";
10: echo "$_POST[message] </p>";
11: //incepe construirea antetului mesajului
12: $msg = "<p><strong>Name:</strong> $_POST[name]</p>";
13: $msg .= "<p><strong>E-Mail:</strong> $_POST[email]</p>";
14: $msg .= "<p><strong>Message:</strong> $_POST[message]</p>";
15: //se configureaza mesajul
16: $recipient = "you@yourdomain.com";
17: $subject = "Form Submission Results";
18: $mailheaders = "MIME-Version: 1.0\r\n";
19: $mailheaders .= "Content-type: text/html; charset=ISO-8859-1\r\n";
20: $mailheaders .= "From: My Web Site <defaultaddress@yourdomain.com> \n";
21: $mailheaders .= "Reply-To: $_POST[email]";
22: //mesajul este trimis
23: mail($recipient, $subject, $msg, $mailheaders);
24: ?>
25: </body>
26: </html>
```

Pe liniile 12–14, şirul de caractere reprezentând mesajul conţine acum cod HTML. Anteturile suplimentare sunt create pe liniile 18–19 unde antetul `MIME-Version` capătă valoarea 1.0, iar antetul `Content-type`, valoarea `text/html` cu setul de caractere `ISO-8859-1`. Atunci când este deschis într-un client de e-mail care acceptă HTML, codul HTML din mesaj va apărea aşa cum a fost intenţionat, după cum se vede în figura 9.8.

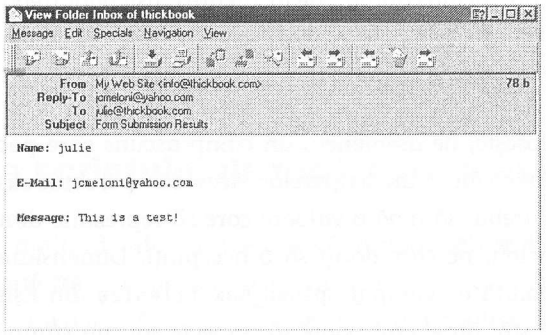


FIGURA 9.8. Mesajul e-mail trimis prin codul din listingul 9.12.

Încărcarea fişierelor pe server

Până acum, ne-am ocupat de forme simple de introducere de date. Totuşi, cele mai populare browsere suportă încărcarea fişierelor pe server (*upload*) şi, evident, acest lucru este acceptat şi de PHP. În această secţiune, vom examina caracteristicile pe care PHP le pune la dispoziţie pentru operaţiunile cu acest tip de date de la utilizator.

Informaţiile despre fişierele încărcate pe server sunt disponibile pentru dumneavoastră prin intermediul vectorului superglobal `$_FILES`, care este indexat după numele câmpurilor de încărcare a fişierelor din formular. Valoarea corespunzătoare fiecăreia dintre aceste chei este, la rândul ei, un vector asociativ. Câmpurile acestor vectori sunt descrise în tabelul 9.1, considerând că numele câmpului din formular care este folosit pentru încărcarea fişierelor are numele `fileupload`.

TABELUL 9.1. Variabilele globale legate de încărcarea fişierelor pe server.

Element	Conţine	Exemplu
`$_FILES['fileupload']['name']`	Numele original al fişierului încărcat pe server.	`test.gif`
`$_FILES['fileupload']['tmp_name']`	Calea către fişierul temporar.	`/tmp/phprDfZvN`
`$_FILES['fileupload']['size']`	Dimensiunea (în octeţi) a fişierului încărcat pe server.	6835
`$_FILES['fileupload']['type']`	Tipul MIME al fişierului încărcat pe server (atunci când este specificat de către client).	`image/gif`

Ţineţi minte elementele acestea câteva momente, până creăm, în secţiunea următoare, formularul de încărcare a fişierelor pe server.

Crearea formularului de încărcare a fişierelor

Mai întâi, trebuie să creăm un formular HTML care să accepte fişierele. Formularele HTML care includ câmpuri de încărcare a fişierelor trebuie să includă un argument ENCTYPE:

```
ENCTYPE="multipart/form-data"
```

Limbajul PHP foloseşte, de asemenea, un câmp ascuns care poate fi inserat înaintea formularului de încărcare a fişierelor. Acest câmp trebuie să aibă numele MAX_FILE_SIZE şi ar trebui să aibă o valoare care să reprezinte dimensiunea de fişier maximă, în octeţi, pe care doriţi să o acceptaţi. Dimensiunea nu poate depăşi valoarea stabilită de câmpul upload_max_filesize din fişierul php.ini care, implicit, este 2MB. Browserul poate să respecte sau nu câmpul MAX_FILE_SIZE, aşa că trebuie să vă bazaţi întotdeauna pe setarea din php.ini pentru a limita dimensiunile fişierelor la o valoare rezonabilă. După ce MAX_FILE_SIZE a fost introdus, sunteţi gata să adăugaţi câmpul de încărcare a fişierului. Acesta este pur şi simplu un element INPUT al cărui argument TYPE este "file". Puteţi să îi daţi ce nume doriţi. Listingul 9.3 pune toate aceste elemente laolaltă într-un formular HTML de încărcare a fişierelor.

LISTINGUL 9.13. Un formular simplu de încărcare a fişierelor.

```
 1: <html>
 2: <head>
 3: <title>Listing 9.13 A simple file upload form</title>
 4: </head>
 5: <body>
 6: <form action="listing9.14.php" enctype="multipart/form-data" method="POST">
 7: <input type="hidden" name="MAX_FILE_SIZE" value="512ØØ">
 8: <p><strong>File to Upload:</strong> <input type="file" name="fileupload"></p>
 9: <p><input type="submit" value="upload!"></p>
1Ø: </form>
11: </body>
12: </html>
```

După cum puteţi vedea, încărcarea fişierelor este limitată la dimensiunea de 50KB pe linia 7, iar numele câmpului de încărcare a fişierelor este fileupload, după cum se vede pe linia 8. Salvaţi acest listing într-un fişier text cu numele listing9.13.html şi amplasaţi fişierul în directorul rădăcină al serverului dumneavoastră Web. Folosiţi browserul pentru a accesa acest formular şi ar trebui să vedeţi ceva similar imaginii din figura 9.9.

Acest formular apelează scriptul listing9.14.php, pe care îl vom crea imediat.

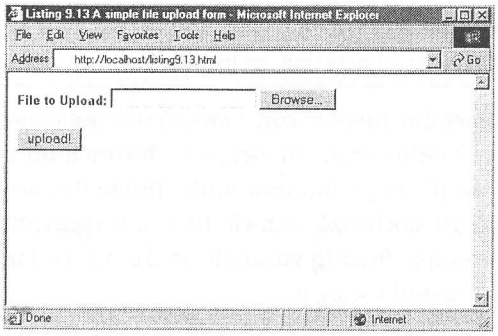


FIGURA 9.9. Formularul creat de codul din listingul 9.13.

Crearea scriptului de încărcare a fişierelor

Dacă vă amintiţi informaţiile despre vectorul superglobal `$_FILES`, atunci aveţi toate informaţiile de care aveţi nevoie pentru a scrie un script simplu de încărcare a fişierelor. Acest script este forţa din spatele formularului creat în listingul 9.13.

LISTINGUL 9.14. Un script de încărcare a fişierelor pe server.

```
 1: <html>
 2: <head>
 3: <title>Listing 9.14 A file upload script</title>
 4: </head>
 5: <body>
 6: <h1>File Upload Results</h1>
 7: <?php
 8: $file_dir = "/path/to/upload/directory";
 9:
10: foreach($_FILES as $file_name => $file_array) {
11:     echo "path: ".$file_array['tmp_name']."<br>\n";
12:     echo "name: ".$file_array['name']."<br>\n";
13:     echo "type: ".$file_array['type']."<br>\n";
14:     echo "size: ".$file_array['size']."<br>\n";
15:
16:     if (is_uploaded_file($file_array['tmp_name'])) {
17:         move_uploaded_file($file_array['tmp_name'],
18:             "$file_dir/$file_array[name]") or die ("Couldn't copy");
19:         echo "file was moved!<br><br>";
20:     }
21: }
22: ?>
23: </body>
24: </html>
```

În listingul 9.14, mai întâi creăm variabila `$file_dir` pe linia 8 pentru a stoca informaţiile despre calea de directoare. Această cale trebuie să fie cea care există pe sistemul dumneavoastră, iar utilizatorul sub care este autentificat serverul Web (de exemplu `www`, `httpd` sau `nobody`) trebuie să aibă permisiunea de a scrie acolo.

Apropo

Calea de directoare folosită pe linia 8 este o cale de tip Unix/Linux. Utilizatorii de Windows trebuie să pună caractere backslash, ca în exemplul: `\My Documents\`.

Pe linia 10 începe o instrucţiune `foreach` care parcurge toate elementele vectorului `$_FILES`. Este folosit acest ciclu iterativ în locul unei instrucţiuni `if` astfel încât scriptul să fie scalabil — să poată prelua mai multe fişiere din aceeaşi pagină. Bucla `foreach` de pe linia 10 stochează numele fişierului încărcat în variabila `$file_name` şi informaţiile despre fişier în variabila `$file_array`. Putem, apoi, să afişăm informaţiile despre fişierul încărcat.

Înainte să mutăm fişierul încărcat din calea sa temporară în calea specificată pe linia 8, verificăm mai întâi dacă fişierul există. Facem acest lucru pe linia 16, folosind funcţia `is_uploaded_file()`. Această funcţie acceptă drept argument o cale de directoare şi returnează valoarea true numai dacă fişierul în chestiune este un fişier valid încărcat. Astfel, această funcţie sporeşte securitatea scripturilor dumneavoastră.

Presupunând că totul este în ordine, fişierul este copiat din calea sa temporară în directorul nou de către codul din liniile 17 şi 18. Folosim o altă funcţie, `move_uploaded_file()` în acest scop. Funcţia copiază un fişier dintr-un loc în altul, efectuând, mai întâi, aceleaşi verificări de securitate ca şi `is_uploaded_file()`. Funcţia `move_uploaded_file()` are nevoie de două argumente — calea fişierului sursă şi calea către destinaţie. Dacă returnează valoarea `true`, mutarea a fost încununată de succes, iar dacă valoarea returnată este `false`, fişierul nu este un fişier valid, sau fişierul nu a putut fi găsit.

Atenţie!

Aveţi grijă la numele fişierelor încărcate. Sistemele de operare cum ar fi Mac OS şi Windows au reguli îngăduitoare de numire a fişierelor, aşa că aşteptaţi-vă ca fişierele să aibă spaţii, semne de citare şi tot felul de caractere neaşteptate. De aceea, este bine să filtraţi numele de fişiere.

Puneţi aceste linii într-un fişier text denumit `listing9.14.html` şi amplasaţi fişierul în directorul rădăcină a serverului dumneavoastră Web. Folosiţi browserul pentru a vă întoarce la formular şi încercaţi să încărcaţi un fişier. Dacă reuşiţi, ar trebui să vedeţi ceva similar cu imaginea din figura 9.10.

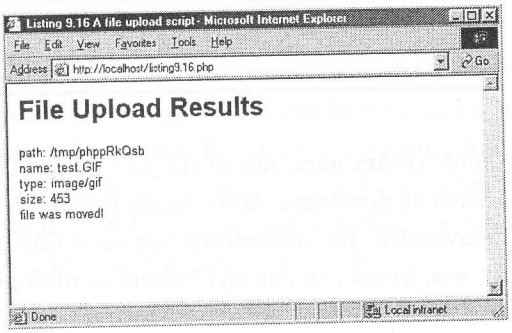


FIGURA 9.10. Un exemplu de rezultat al codului din listingul 9.14.

Rezumat

Lucrurile au început să devină incitante! Bineînţeles, mai lipsesc încă destule. Acum că puteţi prelua informaţii de la utilizator, ar fi drăguţ să şi faceţi ceva cu acestea. Să le scrieţi într-un fişier, eventual. Acesta este subiectul capitolului următor.

De-a lungul acestui capitol, aţi învăţat să lucraţi cu diverse variabile superglobale şi cu datele introduse în formulare. Aţi învăţat, de asemenea, să trimiteţi anteturi brute către client pentru a redirecţiona browserul. Aţi aflat cum să preluaţi informaţii din formularele transmise şi cum să pasaţi informaţii de la o apelare la alta a scriptului cu ajutorul câmpurilor ascunse. În final, aţi învăţat să trimiteţi rezultatele unui formular prin e-mail, precum şi să încărcaţi fişiere din browserul Web prin intermediul unui script PHP.

Atelier

Atelierul este conceput să anticipeze posibile întrebări, să vă ajute să recapitulaţi cele învăţate şi să începeţi să vă puneţi cunoştinţele în practică.

Test

1. Pe care dintre variabilele predefinite o utilizaţi pentru a descoperi numele scriptului?
2. Care dintre vectorii asociativi predefiniţi conţine toate valorile transmise printr-o solicitare de tip `POST`?
3. Care dintre vectorii asociativi predefiniţi conţine toate valorile transmise în cadrul încărcării unui fişier pe server?
4. Ce funcţie folosiţi pentru a redirecţiona browserul spre o pagină nouă?
5. Care sunt cele patru argumente folosite de funcţia `mail()`?
6. Cum limitaţi, de partea clientului, dimensiunea unui fişier pe care utilizatorul îl poate trimite prin intermediul unui anumit formular de încărcare a fişierelor?

Soluţii

1. Variabila `$_SERVER[' PHP_SELF ']` conţine numele scriptului.
2. Vectorul superglobal `$_POST`.
3. Vectorul superglobal `$_FILES`.
4. Funcţia `header()`, împreună cu o adresă.
5. Destinatarul, subiectul, şirul de caractere reprezentând mesajul şi anteturi adiţionale.
6. Folosiţi un câmp ascuns denumit `MAX_FILE_SIZE` în cadrul formularului dumneavoastră.

Activităţi

1. Creaţi calculator aritmetic care să îi permită utilizatorului să trimită două numere şi să aleagă o operaţie (adunare, scădere, înmulţire sau împărţire), iar aceasta să fie efectuată.
2. Folosiţi câmpuri ascunse împreună cu scriptul creat la activitatea 1 pentru a stoca şi afişa numărul de cereri pe care le-a trimis utilizatorul.

CAPITOLUL 10
Lucrul cu sesiunile de utilizator

PHP include numeroase funcţii pentru gestionarea sesiunilor de utilizator, ce pot fi stocate în variabila superglobală `$_SESSION`. Sesiunile folosesc tehnici implementate în limbajul PHP astfel încât salvarea stării devine la fel de uşoară ca apelarea unei funcţii.

În acest capitol, veţi învăţa:

- ce sunt variabilele de sesiune şi cum funcţionează acestea;
- cum să iniţiaţi sau să reluaţi o sesiune;
- cum să stocaţi variabile în cadrul unei sesiuni;
- cum să distrugeţi o sesiune;
- cum să dealocaţi variabilele de sesiune.

Privire de ansamblu asupra funcţiilor dedicate sesiunilor

Funcţiile dedicate sesiunilor ne pun la dispoziţie un identificator unic pentru fiecare utilizator ce poate fi, apoi, folosit pentru preluarea şi stocarea informaţiilor asociate acelui ID. Atunci când un vizitator accesează o pagină pe care funcţionează sesiunile, acesta fie primeşte un nou identificator, fie îi este re-asociat cel pe care l-a avut atunci când a mai accesat respectiva pagină.

Atunci când folosiţi sesiuni, cookie-urile sunt folosite automat pentru a stoca identificatorul de sesiune, dar puteţi să vă asiguraţi că toţi clienţii au acces codificând ID-ul de sesiune în toate linkurile în paginile dumneavoastră bazate pe sesiuni.

Starea unei sesiuni este, de obicei, stocată într-un fişier temporar, deşi puteţi să implementaţi stocarea acesteia într-o bază de date folosind o funcţie cu numele `session_set_save_handler()`. Folosirea funcţiei `session_set_save_handler()` nu face parte din subiectul acestei cărţi, dar puteţi găsi mai multe informaţii la adresa `http://www.php.net/session-set-save-handler`.

Iniţierea unei sesiuni

Ca să lucraţi cu o sesiune, trebuie să iniţiaţi sau să reluaţi sesiunea în mod explicit, în afara cazului în care aţi făcut o modificare în fişierul de configurare `php.ini`. Sesiunile nu sunt iniţiate automat. Dacă vreţi, totuşi, ca ele să fie iniţiate automat, va trebui să găsiţi următoarea linie în `php.ini` şi să îi modificaţi valoarea de la `Ø` la `1` (şi să reporniţi serverul Web):

```
session.auto_start = Ø
```

Schimbând valoarea lui `session.auto_start` la `1`, vă asiguraţi că este iniţiată o sesiune pentru fiecare document PHP. Dacă nu vreţi să schimbaţi acest element de configurare, va trebui să apelaţi funcţia `session_start()` în fiecare script.

După ce a fost iniţiată sesiunea, puteţi avea imediat acces la ID-ul utilizatorului prin intermediul funcţiei `session_id()`. Această funcţie vă permite să aflaţi sau să schimbaţi un ID de sesiune. Listingul 10.1 iniţiază o sesiune şi afişează ID-ul către browser.

LISTINGUL 10.1. Iniţierea sau reluarea unei sesiuni.

```
 1: <?php
 2: session_start();
 3: ?>
 4: <html>
 5: <head>
 6: <title>Listing 1Ø.1 Starting or resuming a session</title>
 7: </head>
 8: <body>
 9: <?php
1Ø: echo "<p>Your session ID is ".session_id()."</p>";
11: ?>
12: </body>
13: </html>
```

Atunci când scriptul este rulat pentru prima oară dintr-un browser, funcţia `session_start()`, apelată pe linia 2, generează un ID de sesiune. Dacă pagina este revizitată sau reîncărcată mai târziu, utilizatorului îi este alocat acelaşi ID de sesiune. Această acţiune presupune ca browserul utilizatorului să accepte cookie-uri. De exemplu, atunci când rulez acest script prima oară, rezultatul este:

```
Your session ID is fa963e3e49186764bØ218e82dØ5Øde7b
```

Când reîncarc pagina, rezultatul este în continuare:

```
Your session ID is fa963e3e49186764bØ218e82dØ5Øde7b
```

deoarece browserul meu acceptă cookie-uri, iar ID-ul de sesiune încă există.

Din moment ce `start_session()` încearcă să scrie un cookie atunci când inițiază prima oară o sesiune, este extrem de important să apelați această funcție înainte să trimiteți orice fel de date către browser. Dacă nu respectați această regulă, sesiunea nu va fi inițiată și, cel mai probabil, veți vedea mesaje de avertizare în pagină.

Sesiunile rămân în vigoare atâta vreme cât browserul este activ. Atunci când utilizatorul repornește browserul, cookie-ul nu mai este păstrat. Puteți schimba acest lucru prin modificarea parametrului `session.cookie_lifetime` în `php.ini`. Valoarea implicită este `0`, dar puteți atribui o valoare în secunde pentru durata de viață a cookie-urilor.

Folosirea variabilelor de sesiune

Accesarea numărului unic de identificare a sesiunii în fiecare dintre documentele dumneavoastră PHP este numai începutul lucrului cu sesiunile. Atunci când este inițiată o sesiune, puteți stoca oricâte variabile în vectorul superglobal `$_SESSION` și, apoi, le puteți accesa din orice pagină care folosește sesiuni.

Apropo

Dacă folosiți o versiune de PHP mai veche decât 4.1.x, vectorul superglobal `$_SESSION` nu există, iar modul de funcționare al sesiunilor este foarte diferit. Dacă nu puteți să actualizați la cea mai nouă versiune de PHP, citiți secțiunea din Manualul PHP referitoare la sesiuni la adresa `http://www.php.net/session`, care include informații despre versiunile mai vechi.

Listingul 10.2 adaugă două variabile în vectorul superglobal `$_SESSION`: `product1` și `product2` (liniile 10 și 11).

LISTINGUL 10.2. Stocarea variabilelor în cadrul unei sesiuni.

```
 1: <?php
 2: session_start();
 3: ?>
 4: <html>
 5: <head>
 6: <title>Listing 10.2 Storing variables in a session</title>
 7: </head>
 8: <body>
 9: <?php
10: $_SESSION[product1] = "Sonic Screwdriver";
11: $_SESSION[product2] = "HAL 2000";
12: echo "The products have been registered.";
13: ?>
14: </body>
15: </html>
```

Puterea magică a listingului 10.2 nu va deveni evidentă decât atunci când utilizatorul trece la o pagină nouă. Listingul 10.3 creează un script PHP separat care accesează variabilele stocate în vectorul superglobal `$_SESSION` de codul din listingul 10.2.

LISTINGUL 10.3. Accesarea variabilelor de sesiune stocate.

```
 1: <?php
 2: session_start();
 3: ?>
 4: <html>
 5: <head>
 6: <title>Listing 10.3 Accessing stored session variables</title>
 7: </head>
 8: <body>
 9: <?php
10: echo "Your chosen products are:";
11: echo "<ul><li>$_SESSION[product1] <li>$_SESSION[product2]\n</ul>\n";
12: ?>
13: </body>
14: </html>
```

Figura 10.1 arată rezultatul listingului 10.3. După cum puteţi vedea, avem acces la variabilele `$_SESSION[product1]` şi `$_SESSION[product2]` dintr-o pagină complet nouă.

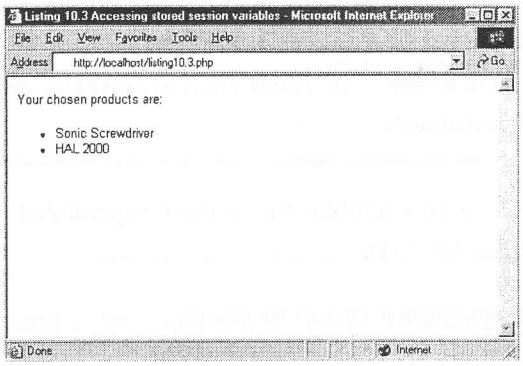


FIGURA 10.1. Accesarea variabilelor de sesiune stocate.

Deşi exemplul nu este prea interesant sau util, scriptul arată cum se accesează variabilele de sesiune stocate. În spatele scenei, PHP scrie informaţiile într-un fişier temporar. Puteţi afla unde anume în sistem este acest fişier apelând funcţia `session_save_pah()`. Această funcţie acceptă drept argument opţional o cale către un director şi, apoi, scrie toate fişierele de sesiune acolo. Dacă nu îi pasaţi nici un argument, ea returnează un şir de caractere reprezentând directorul în care fişierele de sesiune sunt salvate în momentul respectiv. Pe sistemul meu:

```
echo session_save_path();
```

afişează /tmp. Dacă arunc un ochi în directorul meu /tmp, găsesc câteva fişiere cu nume precum:

```
sess_fa963e3e49186764b0218e82d050de7b
sess_76cae8ac1231b11afa2c69935c11dd95
sess_bb50771a769c605ab77424d59c784ea0
```

Deschizând fişierul care corespunde cu ID-ul ce mi-a fost alocat când am rulat pentru prima oară listingul 10.1, pot vedea cum au fost stocate variabilele înregistrate:

```
product1|s:17:"Sonic Screwdriver";product2|s:8:"HAL 2000";
```

Atunci când o variabilă este plasată în vectorul superglobal $_SESSION, PHP scrie numele variabilei şi valoarea acesteia într-un fişier. Aceste informaţii pot, apoi, să fie citite astfel încât variabilele să fie înviate mai târziu. După ce adăugaţi o variabilă în vectorul superglobal $_SESSION, îi puteţi schimba valoarea în orice moment al execuţiei scriptului, dar modificarea valorii nu va fi reflectată la nivel global decât după ce reatribuiţi variabila vectorului superglobal $_SESSION.

Exemplul din listingul 10.2 demonstrează procesul de adăugare a variabilelor în vectorul superglobal $_SESSION. Acest exemplu, însă, nu este foarte flexibil. În mod ideal, ar trebui să puteţi înregistra un număr variabil de valori. Aţi putea să vreţi să le daţi utilizatorilor posibilitatea de a alege produse dintr-o listă, de exemplu. În acest caz, puteţi folosi funcţia serialize() pentru a stoca un vector în cadrul sesiunii dumneavoastră.

Listingul 10.4 creează un formular care îi permite utilizatorului să aleagă mai multe produse. În acest caz, trebuie să puteţi folosi variabilele de sesiune pentru a crea un coş de cumpărături virtuale rudimentar.

LISTINGUL 10.4. Adăugarea unei variabile de tip vector într-o variabilă de sesiune.

```
<?php
session_start();
?>
<html>
<head>
<title>Listing 10.4 Storing an array with a session</title>
</head>
<body>
<h1>Product Choice Page</h1>
<?php
if (isset($_POST[form_products])) {
    if (!empty($_SESSION[products])) {
        $products = array_unique(
        array_merge(unserialize($_SESSION[products]),
        $_POST[form_products]));
        $_SESSION[products] = serialize($products);
    } else {
        $_SESSION[products] = serialize($_POST[form_products]);
  }
   echo "<p>Your products have been registered!</p>";
```

LISTINGUL 10.4. (*continuare*)

```
21: }
22: ?>
23: <form method="POST" action="<?php $_SERVER[PHP_SELF] ?>">
24: <P><strong>Select some products:</strong><br>
25: <select name="form_products[]" multiple size=3>
26: <option value="Sonic Screwdriver">Sonic Screwdriver</option>
27: <option value="Hal 2000">Hal 2000</option>
28: <option value="Tardis">Tardis</option>
29: <option value="ORAC">ORAC</option>
30: <option value="Transporter bracelet">Transporter bracelet</option>
31: </select>
32:
33: <P><input type="submit" value="choose"></p>
34: </form>
35:
36: <p><a href="listing10.5.php">go to content page</a></p>
37: </body>
38: </html>
```

Începem prin a iniţia sau relua o sesiune apelând funcţia `session_start()` pe linia 2. Acest lucru ar trebui să ne asigure accesul la orice variabile de sesiune alocate anterior. Începem să scriem un formular HTML pe linia 23 şi, pe linia 25, creăm un element de tip `SELECT` denumit `form_products[]`, care conţine elemente `OPTION` pentru un număr de produse. Reţineţi că elementele formularului HTML care permit selecţie multiplă trebuie să aibă o pereche de paranteze pătrate adăugate la valoarea argumentului NAME. Acest lucru permite accesarea alegerilor utilizatorului prin intermediul unui vector.

În cadrul blocului de cod PHP care începe pe linia 10, verificăm prezenţa vectorului `$_POST[form_products]` (linia 11). Dacă variabila este prezentă, putem presupune că formularul a fost trimis şi că informaţiile au fost deja stocate în vectorul superglobal `$_SESSION`. După aceasta, verificăm, pe linia 12, dacă există un vector cu numele `$_SESSION[products]`. Dacă vectorul există, acesta a fost populat cu ocazia unei vizite anterioare, aşa că în concatenăm cu vectorul `$_POST[form_products]`, extragem elementele unice şi atribuim rezultatul vectorului `$products` (liniile 13–15). Apoi adăugăm vectorul `$products` ca element în `$_SESSION` pe linia 16.

Linia 36 conţine un link către un alt script, pe care îl vom folosi pentru a demonstra faptul că avem acces la produsele pe care utilizatorul le-a ales. Creăm acest nou script în listingul 10.5.

LISTINGUL 10.5. Accesarea variabilelor de sesiune.

```
1: <?php
2: session_start();
3: ?>
4: <html>
5: <head>
6: <title>Listing 10.5 Accessing session variables</title>
```

LISTINGUL 10.5. (*continuare*)

```
 7: </head>
 8: <body>
 9: <h1> Content Page</h1>
10: <?php
11: if (isset($_SESSION[products])) {
12:    echo "<strong>Your cart:</strong><ol>";
13:    foreach (unserialize($_SESSION[products]) as $p) {
14:        echo "<li>$p";
15:    }
16:    echo "</ol>";
17: }
18: ?>
19: <p><a href="listing10.4.php">return to product choice page</a></p>
20: </body>
21: </html>
```

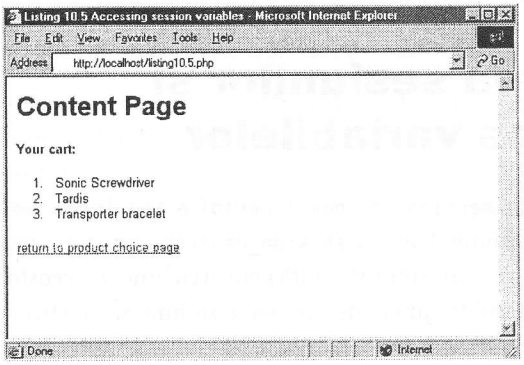


FIGURA 10.2. Accesarea unui vector de variabile de sesiune.

Din nou, folosim, pe linia 2, `session_start()` pentru a relua sesiunea. Verificăm prezenţa variabilei `$_SESSION[products]` pe linia 11. Dacă aceasta există, folosim funcţia `unserialize()` şi parcurgem element cu element vectorul pe liniile 13–15, afişând în browser fiecare dintre obiectele alese de către utilizator. Un exemplu poate fi văzut în figura 10.2.

Pentru programarea unui coş de cumpărături virtual de-adevărat, va trebui, bineînţeles, să aveţi detaliile despre produse stocate într-o bază de date şi să verificaţi datele introduse de utilizator în loc să le salvaţi şi să le afişaţi orbeşte, dar listingurile 10.4 şi 10.5 demonstrează uşurinţa cu care puteţi folosi funcţiile de sesiune pentru a accesa variabile de tip vector create pe alte pagini.

Pasarea ID-urilor de sesiune prin intermediul şirurilor de interogare

Până acum v-aţi bazat pe cookie-uri pentru a salva ID-urile de sesiune între apelările diverselor scripturi. În sine, metoda aceasta nu este cea mai demnă de

încredere deoarece nu puteţi fi sigur că browserul acceptă cookie-uri. Ca să implementaţi o măsură de siguranţă, totuşi, puteţi pasa ID-ul de sesiune de la un script la altul înglobându-l într-un şir de interogare. PHP vă pune la dispoziţie o pereche nume / valoare într-o variabilă denumită SID în cazul în care nu este găsit un cookie cu valoarea ID-ului de sesiune. Puteţi adăuga acest şir în oricare dintre linkurile HTML dintr-o pagină care foloseşte sesiuni:

```
<a href="anotherpage.html?<?php echo SID; ?>">Alta pagina</a>
```

Acest link va ajunge la browser sub forma:

```
<a href="anotherpage.html?PHPSESSID=08ecedf79fe34561fa82591401a01da1">Alta pagina</a>
```

ID-ul de sesiune pasat în acest fel va fi recunoscut automat de pagina ţintă atunci când este apelată funcţia `session_start()`, iar dumneavoastră veţi avea acces la variabilele de sesiune în mod normal.

Distrugerea sesiunilor şi dealocarea variabilelor

Puteţi folosi funcţia `session_destroy()` pentru a termina o sesiune, ştergând toate variabilele de sesiune. Funcţia `session_destroy()` nu are nevoie de nici un argument. Trebuie să aveţi o sesiune stabilită pentru ca funcţia aceasta să funcţioneze corespunzător. Următorul fragment de cod reia o sesiune şi o distruge brusc:

```
session_start();
session_destroy();
```

Atunci când treceţi la o altă pagină care lucrează cu sesiuni, sesiunea pe care aţi distrus-o nu mai este disponibilă, forţând pagina respectivă să iniţieze o nouă sesiune. Orice variabile înregistrate vor fi pierdute.

Funcţia `session_destroy()` nu distruge imediat toate variabilele înregistrate. Ele rămân accesibile scriptului în care a fost apelată funcţia `session_destroy()` (până când acesta este reîncărcat). Fragmentul de cod de mai jos reia sau iniţiază o sesiune şi înregistrează o variabilă denumită test, căreia îi atribuim valoarea 5. Distrugând sesiunea, nu distrugem şi variabila înregistrată.

```
session_start();
$_SESSION[test] = 5;
session_destroy();
print $_SESSION[test]; // afiseaza 5
```

Pentru a elimina toate variabilele înregistrate dintr-o sesiune, trebuie pur şi simplu să dealocaţi variabilele respective, folosind funcţia `unset()`:

```
session_start();
$_SESSION[test] = 5;
session_destroy();
unset($_SESSION[test]);
print $_SESSION[test]; // nu afiseaza nimic.
```

Rezumat

În acest capitol, am analizat diverse moduri de a salva starea într-un protocol care nu conţine conceptul de stare. Toate metodele folosesc o combinaţie de şiruri de interogare şi cookie-uri, uneori în conjuncţie cu folosirea fişierelor sau a bazelor de date. Toate aceste abordări au, fiecare, avantajele şi defectele proprii.

Aţi învăţat că un cookie nu este, în sine, o metodă de încredere şi că nu poate stoca prea multe informaţii. Pe de altă parte, cookie-urile pot persista pe perioade lungi de timp. Abordările care se bazează pe scrierea informaţiilor într-un fişier sau într-o bază de date necesită costuri pentru a se putea asigura viteza de acces necesară şi pot pune probleme serioase în cazul site-urilor foarte populare. Cu toate acestea, un simplu ID poate reprezenta cheia către cantităţi importante de date stocate pe disc. Pentru a vă asigura că marea majoritate a utilizatorilor vor putea beneficia de modul de lucru bazat pe sesiuni, puteţi folosi constanta SID ca să pasaţi ID-ul de sesiune ca parte a şirului de interogare.

În ceea priveşte sesiunile în sine, aţi învăţat cum să iniţializaţi sau să reluaţi o sesiune cu ajutorul funcţiei `session_start()`. În cadrul unei sesiuni, aţi învăţat să adăugaţi variabile la vectorul superglobal `$_SESSION`, să verificaţi dacă acestea există, să le dealocaţi atunci când vreţi şi să distrugeţi întreaga sesiune.

Întrebări şi răspunsuri

Î. Ar trebui să fiu conştient de anumite capcane ale folosirii funcţiilor de sesiune?

R. Funcţiile de sesiune sunt, de obicei, de încredere. Cu toate acestea, reţineţi că cookie-urile nu pot fi citite de către domenii diferite, aşa că, dacă proiectul dumneavoastră utilizează mai multe nume de domeniu pe acelaşi server (poate în cadrul unui mediu de e-comerţ), s-ar putea să trebuiască să luaţi în considerare dezactivarea cookie-urilor de sesiune dând valoarea Ø directivei `session.use_cookies` din fişierul `php.ini`.

Atelier

Atelierul este conceput să anticipeze posibile întrebări, să vă ajute să recapitulaţi cele învăţate şi să începeţi să vă puneţi cunoştinţele în practică.

Test

1. Ce funcţie aţi folosi pentru a iniţia sau relua o sesiune?
2. Ce funcţie returnează ID-ul sesiunii curente?
3. Cum încheiaţi o sesiune şi ştergeţi orice urmă a acesteia la vizitele ulterioare?
4. Ce returnează constanta SID?

Soluţii

1. Puteţi iniţia o sesiune folosind funcţia session_start().
2. Puteţi accesa ID-ul sesiunii folosind funcţia session_id().
3. Funcţia `session_destroy()` elimină orice urme ale unei sesiuni pentru apelurile ulterioare.
4. Dacă nu sunt disponibile cookie-urile, constanta SID conţine o pereche nume / valoare care poate fi încorporată într-un şir de interogare. Acesta va pasa ID-ul de sesiune de la o apelare a unui script la alta.

Activitate

Creaţi un script care să folosească funcţii de sesiune pentru a ţine minte câte pagini de pe serverul dumneavoastră a vizitat un utilizator. Oferiţi-i utilizatorului o listă cu linkuri către fiecare dintre paginile vizitate astfel încât să îi fie uşor să reconstituie „drumul" parcurs prin site.

CAPITOLUL 11
Manipularea fişierelor şi directoarelor

Verificarea, citirea şi scrierea fişierelor reprezintă activităţi esenţiale pentru orice limbaj de programare. PHP nu face excepţie, punând la dispoziţie funcţii care permit un proces simplu şi direct.

În acest capitol, veţi învăţa cum să:

- includeţi fişiere în documentele dumneavoastră;
- verificaţi existenţa fişierelor şi a directoarelor;
- deschideţi un fişier înainte să lucraţi cu el;
- citiţi date dintr-un fişier;
- scrieţi sau să adăugaţi într-un fişier;
- blocaţi un fişier;
- lucraţi cu directoarele.

Includerea fişierelor cu ajutorul instrucţiunii `include()`

Instrucţiunea `include()` vă permite să încorporaţi fişiere în documentele dumneavoastră PHP. Codul PHP din aceste fişiere poate fi executat ca şi când ar fi parte din documentul principal. Acest lucru poate fi util pentru includerea de biblioteci de cod în mai multe pagini.

După ce creaţi o funcţie foarte utilă, singura opţiune pe care o aveaţi până acum era să o copiaţi în fiecare document care ar fi avut nevoie de ea. De bună seamă, dacă descopereaţi o greşeală sau doreaţi să adăugaţi ceva funcţiei respective, ar fi trebuit să găsiţi fiecare pagină care o folosea şi să operaţi peste tot modificările. Instrucţiunea `include()` poate să vă scape din această situaţie. Puteţi pune funcţia într-un sigur fişier şi, la momentul rulării, să o încărcaţi în orice pagină care are nevoie de ea. Instrucţiunea `include()` are nevoie de un singur argument: o cale relativă către fişierul care trebuie inclus. Listingul 11.1 creează un script PHP simplu care foloseşte `include()` pentru a încorpora şi afişa conţinutul unui fişier.

LISTINGUL 11.1. Folosirea instrucțiunii `include()`.

```
 1: <html>
 2: <head>
 3: <title>Listing 11.1 Using include()</title>
 4: </head>
 5: <body>
 6: <?php
 7: include("listing11.2.php");
 8: ?>
 9: </body>
10: </html>
```

Instrucțiunea `include()` din listingul 11.1 încorporează documentul din fișierul `listing11.2.php`, al cărui conținut îl puteți vedea în listingul 11.2.

LISTINGUL 11.2. Fișierul inclus în listingul 11.1.

```
1: I have been included!!
```

Introduceți conținutul listingului 11.1 într-un fișier numit `listing11.1.php`, și conținutul listingului 11.2 într-un fișier cu numele `listing11.2.php`. Amplasați ambele fișiere în directorul rădăcină al serverului dumneavoastră Web. Atunci când accesați în browser fișierul `listing11.1.php`, rezultatul afișat pe ecran va fi:

```
I have been included!!
```

Acest lucru vi se poate părea ciudat, dat fiind faptul că am inclus text simplu într-un bloc de cod PHP. De fapt, conținutul unui fișier inclus este afișat implicit sub formă de text. Dacă vreți să executați cod PHP dintr-un fișier inclus, trebuie să puneți codul din fișierul inclus între marcajele de început și sfârșit al codului PHP. În listingurile 11.3 și 11.4, am modificat exemplul precedent astfel încât codul din fișierul inclus să fie executat.

LISTINGUL 11.3. Folosirea instrucțiunii `include()` pentru executarea codului PHP dintr-un alt fișier.

```
 1: <html>
 2: <head>
 3: <title>Listing 11.3 Using include to execute PHP in another file</title>
 4: </head>
 5: <body>
 6: <?php
 7: include("listing11.4.php");
 8: ?>
 9: </body>
10: </html>
```

LISTINGUL 11.4. Un fişier inclus conţinând cod PHP.

```
1: <?php
2: echo "I have been included!!<BR>";
3: echo "But now I can add up...  4 + 4 = ".(4 + 4);
4: ?>
```

Puneţi conţinutul listingului 11.3 într-un fişier denumit `listing11.3.php` şi conţinutul listingului 11.4 într-un fişier denumit `listing11.4.php`. Amplasaţi ambele fişiere în directorul rădăcină al serverului dumneavoastră Web. Atunci când accesaţi în browser fişierul `listing11.3.php`, rezultatul afişat pe ecran va fi:

```
I have been included!!
But now I can add up... 4 + 4 = 8
```

Returnarea unei valori dintr-un document inclus

Fişierele incuse în PHP pot returna valori în acelaşi fel în care o fac funcţiile. Ca şi în cazul unei funcţii, folosirea instrucţiunii return opreşte execuţia codului din fişierul inclus. Mai mult, nu se mai include nici codul HTML care urmează. În listingurile 11.5 şi 11.6 am inclus un fişier şi am atribuit unei variabile valoarea returnată de acesta.

LISTINGUL 11.5. Folosirea instrucţiunii `include()` pentru a executa cod PHP şi a atribui unei variabile valoarea returnată.

```
 1: <html>
 2: <head>
 3: <title>Listing 11.5 Using include() to execute PHP and
 4:     assign the return value</title>
 5: </head>
 6: <body>
 7: <?php
 8: $addResult = include("listing11.6.php");
 9: echo "The include file returned $addResult";
10: ?>
11: </body>
12: </html>
```

LISTINGUL 11.6. Un fişier inclus care returnează o valoare.

```
1: <?php
2: $retval = ( 4 + 4 );
3: return $retval;
4: ?>
5: Acest cod HTML nu va fi afişat niciodată, deoarece succede o instrucţiune
return!
```

Puneţi conţinutul listingului 11.5 într-un fişier denumit `listing11.5.php` şi conţinutul listingului 11.6 într-un fişier denumit `listing11.6.php`. Amplasaţi ambele fişiere în directorul rădăcină al serverului dumneavoastră Web. Atunci când accesaţi în browser fişierul `listing11.5.php`, rezultatul afişat pe ecran va fi:

```
The include file returned 8
```

Folosirea instrucţiunii `include()` în cadrul structurilor de control

Puteţi folosi o instrucţiune `include()` în cadrul unei instrucţiuni condiţionale, iar fişierul respectiv va fi inclus doar în cazul în care condiţia este îndeplinită. De exemplu, instrucţiunea `include()` din fragmentul de cod de mai jos nu va fi apelată niciodată:

```
$test = false;
if ($test) {
   include("a_file.txt"); // nu va fi de fapt inclus
}
```

Dacă folosiţi o instrucţiune `include()` în cadrul unei structuri repetitive, ea va fi înlocuită la fiecare apel cu conţinutul fişierului referit. Acest conţinut este executat la fiecare apel. Listingul 11.7 ilustrează acest concept folosind o instrucţiune `include()` în cadrul unei bucle `for`. Instrucţiunea `include()` face referire la un fişier diferit la fiecare iteraţie.

LISTINGUL 11.7. Folosirea instrucţiunii `include()` în cadrul unei bucle.

```
 1: <html>
 2: <head>
 3: <title>Listing 11.7 Using include() within a loop</title>
 4: </head>
 5: <body>
 6: <?php
 7: for ($x = 1; $x<=3; $x++) {
 8:    $incfile = "incfile$x".".txt";
 9:    echo "Attempting include $incfile<br>";
10:    include("$incfile");
11:    echo "<br>";
12: }
13: ?>
14: </body>
15: </html>
```

Atunci când listingul 11.7 este rulat, el include conţinutul a trei fişiere diferite: `incfile1.txt`, `incfile2.txt` şi `incfile3.txt`. Presupunând că fiecare dintre aceste fişiere conţine pur şi simplu o confirmare a numelui său, rezultatul ar trebui să arate ca în figura 11.1.

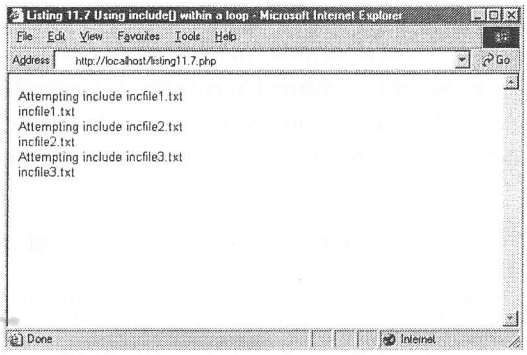


FIGURA 11.1. Rezultatul listingului 11.7.

Instrucţiunea `include_once()`

Una dintre problemele create de folosirea multor biblioteci de cod în scripturile dumneavoastră este pericolul includerii aceluiaşi fişier de mai multe ori. Acest lucru se poate petrece în cazul proiectelor mari, când diferite biblioteci includ fiecare acelaşi fişier. Includerea unui fişier de două ori duce la declararea repetată a funcţiilor şi a claselor, fapt ce îi provoacă multe necazuri motorului de PHP.

Situaţia este rezolvată de instrucţiunea `inlcude_once()`. Această instrucţiune are nevoie de calea către un fişier de inclus şi, când este apelată prima oară, se comportă similar cu instrucţiunea `include()`. Totuşi, dacă instrucţiunea `include_once()` este apelată din nou pentru acelaşi fişier în timpul execuţiei unui script, fişierul nu mai este inclus încă o dată. Iată de ce `include_once()` este un instrument excelent pentru crearea de biblioteci de cod reutilizabile!

Directiva `include_path`

Folosind instrucţiunile `include()` şi `include_once()` pentru a accesa bibliotecile puteţi spori flexibilitatea şi reutilizabilitatea proiectelor dumneavoastră. Cu toate acestea, mai există câteva probleme ce vă pot da dureri de cap. Portabilitatea, în special, poate fi diminuată în cazul în care codaţi static căile către fişierele incluse. Imaginaţi-vă că aţi creat un director `lib` şi că faceţi referire la acesta peste tot în proiectul dumneavoastră:

```
include_once("/home/user/bob/htdocs/project4/lib/mylib.inc.php");
```

Atunci când mutaţi proiectul pe un server nou, s-ar putea să descoperiţi că aveţi de modificat o sută şi ceva de instrucţiuni include. Puteţi scăpa de această soartă crudă utilizând directiva `include_path` în fişierul `php.ini`:

```
include_path   .:/home/user/bob/htdocs/project4/lib/
```

Directiva `include_path` poate conține oricâte directoare vreți, separate prin caracterul două puncte (sau punct și virgulă, în Windows). Ordinea elementelor din directiva `include_path` determină ordinea în care fișierul respectiv este căutat în directoare. Primul punct (`.`) dinaintea primului caracter două puncte semnfică „directorul curent". Puteți, apoi, să faceți referire la un fișier bibliotecă folosind doar numele acestuia:

```
include_once("mylib.inc.php");
```

Atunci când mutați proiectul, nu trebuie să schimbați decât directiva `include_path`.

Apropo

PHP are atât o instrucțiune `require()`, care are un efect similar cu `include()`, cât și o instrucțiune `require_once()`.

Instrucțiunea `require()` se execută indiferent de curgerea scriptului și, de aceea, nu trebuie folosită ca parte dintr-o buclă sau instrucțiune condițională.

Un fișier inclus cu instrucțiunea `require()` nu poate returna o valoare.

Verificarea fișierelor

Înainte de a lucra cu un fișier sau director, este, de cele mai multe ori, o idee bună să aflați câte ceva despre acesta. PHP vă pune la dispoziție multe funcții care vă ajută să descoperiți informații despre fișierele de pe sistemul dumneavoastră. Secțiunea aceasta discută pe scurt câteva dintre cele mai utile astfel de funcții.

Verificarea existenței unui fișier cu ajutorul funcției `file_exists()`

Puteți verifica existența unui fișier folosind funcția `file_exists()`. Această funcție are nevoie de un argument de tip șir de caractere ce reprezintă o cale absolută sau relativă către un fișier care ar putea sau nu să fie acolo. Dacă fișierul este găsit, `file_exists()` returnează valoarea `true`; altminteri, ea returnează `false`.

```
if (file_exists("test.txt")) {
        echo "The file exists!";
}
```

Este fișier sau director?

Puteți să vă asigurați dacă entitatea pe care o verificați este un fișier și nu un director, cu ajutorul funcției `is_file()`. Acesteia trebuie să i se paseze o cale către un fișier și returnează o valoare de tip `Boolean`.

```
if (is_file("test.txt")) {
        echo "test.txt is a file!";
}
```

Alternativ, puteţi să verificaţi dacă entitatea pe care o verificaţi este un director. Puteţi face acest lucru cu ajutorul funcţiei `is_dir()`, căreia trebuie să îi pasaţi o cale către respectivul director şi returnează o valoare de tip `Boolean`.

```
if (is_dir("/tmp")) {
        echo "/tmp is a directory";
}
```

Verificarea stării unui fişier

O dată ce aţi aflat că un fişier există şi că este ceea ce vă aşteptaţi să fie, puteţi să aflaţi câteva dintre lucrurile pe care le puteţi face cu el. În mod obişnuit, aţi putea să vreţi să îl citiţi, să scrieţi în el sau să îl executaţi. PHP vă poate ajuta la toate aceste operaţiuni.

Funcţia `is_readable()` vă spune dacă puteţi sau nu să citiţi un fişier. Pe sistemele Unix este posibil să aveţi voie să vedeţi un fişier, dar să vă fie interzisă citirea conţinutului acestuia. Funcţia necesită drept argument calea către un fişier şi returnează o valoarea de tip `Boolean`.

```
if (is_readable("test.txt")) {
        echo "test.txt is readable";
}
```

Funcţia `is_writable()` vă spune dacă puteţi să scrieţi într-un fişier. La fel ca şi în cazul funcţiei `is_readable()`, şi `is_writable()` necesită drept argument calea către fişier şi returnează o valoare de tip `Boolean`.

```
if (is_writable("test.txt")) {
        echo "test.txt is writable";
}
```

Funcţia `is_executable()` vă spune dacă puteţi sau nu să rulaţi un fişier, bazându-se fie pe privilegiile fişierului, fie pe extensia sa, în funcţie de platformă. Funcţia acceptă calea către fişier şi returnează o valoare de tip `Boolean`.

```
if (is_executable("test.txt")) {
        echo "test.txt is executable";
}
```

Determinarea dimensiunii unui fişier cu ajutorul funcţiei `filesize()`

Dată fiindu-i calea către un fişier, funcţia `filesize()` încearcă să determine şi să returneze dimensiunea sa în octeţi. Funcţia returnează valoarea `false` dacă întâlneşte probleme.

```
echo "The size of test.txt is.. ";
echo filesize("test.txt");
```

Obținerea informațiilor despre data fișierului

Uneori trebuie să ştiţi când s-a scris sau s-a citit ultima oară dintr-un fişier. PHP are câteva funcţii care pot să ne furnizeze astfel de informaţii.

Puteţi afla când a fost accesat ultima oară un fişier cu ajutorul funcţiei `fileatime()`. Această funcţie are nevoie de calea către fişier şi returnează data la care fişierul a fost accesat ultima oară. A accesa un fişier înseamnă fie a citi, fie a scrie în el. Datele returnate de toate funcţiile de acest tip sunt în formatul unui *timestamp* — numărul de secunde scurse de la 1 ianuarie 1970. În exemplele noastre, folosim funcţia `date()` pentru a traduce această valoare într-o formă inteligibilă pentru oameni.

```
$atime = fileatime("test.txt");
echo "test.txt was last accessed on ";
echo date("D d M Y g:i A", $atime);
// Exemplu de rezultat: Mon 11 Aug 2003 9:54 PM
```

Puteţi descoperi data ultimei modificări aduse unui fişier folosind funcţia `filemtime()`, care are nevoie de calea către fişier şi returnează data la care fişierul a fost modificat ultima oară în formatul bazat pe epoca Unix. A modifica un fişier înseamnă a-i schimba conţinutul în orice fel.

```
$mtime = filemtime("test.txt");
echo "test.txt was last modified on ";
echo date("D d M Y g:i A", $mtime);
// Exemplu de rezultat: Mon 11 Aug 2003 9:54 PM
```

PHP vă mai permite şi să verificaţi data ultimei schimbări suferite de fişier prin intermediul funcţiei `filectime()`. Pe un sistem Unix, data schimbării reprezintă momentul în care fişierului i-a fost modificat ultima oară conţinutul sau i s-au schimbat privilegiile de acces sau proprietarul. Pe alte platforme, `filectime()` returnează data creării fişierului.

```
$ctime = filectime("test.txt");
echo "test.txt was last changed on ";
echo date("D d M Y g:i A", $ctime);
// Exemplu de rezultat: Mon 11 Aug 2003 9:54 PM
```

Crearea unei funcții care efectuează mai multe verificări asupra fișierelor

Listingul 11.8 creează o funcţie ce pune la un loc, într-un singur script, toate funcţiile de verificare a fişierelor pe care le-am discutat până acum.

LISTINGUL 11.8. O funcție care afișează rezultatele mai multor verificări asupra unui fișier.

```
1: <html>
2: <head>
3: <title>Listing 11.8 A function to output the results of
4:     multiple file tests</title>
5: </head>
6: <body>
```

LISTINGUL 11.8. (*continuare*)

```
 7: <?php
 8: $file = "test.txt";
 9: outputFileTestInfo($file);
10:
11: function outputFileTestInfo($f) {
12:    if (!file_exists( $f)) {
13:        echo "<p>$f does not exist</p>";
14:        return;
15:    }
16:    echo "<p>$f is ".(is_file($f)?"":"not ")."a file</p>";
17:    echo "<p>$f is ".(is_dir($f)?"":"not ")."a directory</p>";
18:    echo "<p>$f is ".(is_readable($f)?"":"not ")."readable</p>";
19:    echo "<p>$f is ".(is_writable($f)?"":"not ")."writable</p>";
20:    echo "<p>$f is ".(is_executable($f)?"":"not ")."executable</p>";
21:    echo "<p>$f is ".(filesize($f))." bytes</p>";
22:    echo "<p>$f was accessed on ".date( "D d M Y g:i A",fileatime($f))."</p>";
23:    echo "<p>$f was modified on ".date( "D d M Y g:i A",filemtime($f))."</p>";
24:    echo "<p>$f was changed on ".date( "D d M Y g:i A",filectime($f) )."</p>";
25: }
26: ?>
27: </body>
28: </html>
```

Dacă aţi salva acest cod în directorul rădăcină al serverului dumneavoastră Web şi l-aţi rula cu ajutorul browserului, rezultatul ar arăta asemănător cu figura 11.2.

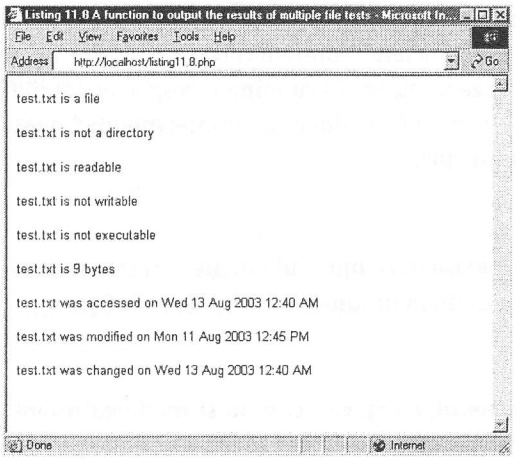


FIGURA 11.2. Rezultatul listingului 11.8.

Remarcaţi că am folosit operatorul ternar pentru a opera în mod compact cu unele dintre aceste teste. Haideţi să analizăm mai în detaliu unul dintre aceste teste — cel care se găseşte pe linia 16:

```
echo "<p>$f is ".(is_file($f)?"":"not ")."a file</p>";
```

Folosim funcţia `is_file()` pe post de expresie de partea stângă a operatorului ternar. Dacă aceasta returnează valoarea `true`, este returnat un şir vid. Altminteri, şirul de caractere returnat este `"not "`. Valoarea returnată de expresia ternară este adăugată şirului care trebuie afişat cu ajutorul operatorilor de concatenare. Această expresie ar putea fi scrisă mai clar, dar mai puţin compact, sub forma următoare:

```
$is_it = is_file($f)?"":"not ";
echo $f." is ".$is_it." a file";
```

Am putea, bineînţeles, să fim şi mai clari folosind o instrucţiune `if`, dar închipuiţi-vă cât de mare ar deveni funcţia dacă am folosi varianta următoare:

```
if (is_file($f)) {
        echo "$f is a file<br>";
} else {
        echo "$f is not a file<br>";
}
```

Din cauză că rezultatul acestor trei abordări este acelaşi, o puteţi alege pe oricare dintre ele în funcţie de preferinţe.

Crearea şi ştergerea fişierelor

Dacă un fişier nu există încă, îl puteţi crea cu ajutorul funcţiei `touch()`. Primind drept argument un şir de caractere reprezentând o cale către un fişier, funcţia `touch()` va încerca să creeze un fişier vid cu numele respectiv. Dacă fişierul există deja, conţinutul său nu este modificat, dar data ultimei modificări este actualizată la momentul executării funcţiei.

```
touch("myfile.txt");
```

Puteţi elimina un fişier existent cu ajutorul funcţiei `unlink()`. La fel ca `touch()`, şi funcţia `unlink()` acceptă drept argument calea către un fişier:

```
unlink("myfile.txt");
```

Toate funcţiile care creează, şterg, citesc, scriu şi modifică fişiere pe sistemele Unix au nevoie ca fişierul sau directorul ţintă să aibă stabilite privilegiile de acces corespunzătoare.

Deschiderea unui fişier pentru scriere, citire sau adăugare

Înainte de a putea lucra cu un fişier, trebuie mai întâi să îl deschideţi pentru citire, scriere sau pentru amândouă. PHP ne pune la dispoziţie, pentru aceste operaţiuni, funcţia `fopen()`. Aceasta are nevoie de un argument de tip şir de caractere

care conţine calea către fişierul respectiv, urmat de un şir de caractere care specifică modul în care trebuie deschis fişierul. Cele mai uzuale moduri sunt `r` pentru citire (read), `w` pentru scriere (write) şi `a` pentru adăugare (append). Funcţia `fopen()` returnează o valoare de tip fişier pe care o veţi utiliza ulterior când veţi lucra cu fişierul respectiv. Ca să deschideţi un fişier pentru citire, veţi folosi următoarea sintaxă:

```
$fp = fopen("test.txt", "r");
```

Ca să deschideţi un fişier pentru scriere, folosiţi următoarea sintaxă:

```
$fp = fopen("test.txt", "w");
```

Ca să deschideţi un fişier pentru adăugare (adică, pentru a adăuga date la sfârşitul fişierului), folosiţi sintaxa următoare:

```
$fp = fopen("test.txt", "a");
```

Funcţia `fopen()` returnează valoarea `false` dacă fişierul nu poate fi deschis, indiferent de motiv. De aceea, este o idee bună să verificaţi valoarea returnată de către funcţie înainte de a trece la lucrul cu fişierul respectiv. Puteţi face acest lucru cu ajutorul unei instrucţiuni `if`:

```
if ($fp = fopen("test.txt", "w")) {
   // do something with $fp
}
```

Sau puteţi folosi un operator logic pentru a termina execuţia în cazul în care este vorba de un fişier esenţial:

```
($fp = fopen("test.txt", "w")) or die ("Couldn't open file, sorry");
```

Dacă `fopen()` returnează valoarea `true`, restul expresiei nu va fi procesat şi nu se va ajunge niciodată la funcţia `die()` (care scrie un mesaj în browser şi opreşte complet scriptul). În caz contrar, partea dreaptă a expresiei or este procesată şi este apelată funcţia `die()`.

Presupunând că totul merge bine şi că treceţi mai departe şi lucraţi cu fişierul, trebuie să ţineţi minte să îl închideţi atunci când terminaţi. Puteţi face acest lucru apelând funcţia `fclose()`, care are nevoie de un argument de tip fişier — returnat de un apel `fopen()` încununat cu succes:

```
fclose($fp);
```

Citirea din fişiere

PHP oferă un număr de funcţii pentru citirea datelor din fişiere. Acestea vă permit să citiţi octet cu octet, linie cu linie sau caracter cu caracter.

Citirea liniilor dintr-un fişier cu `fgets()` şi `feof()`

După ce deschideţi un fişier pentru citire, de multe ori aveţi nevoie să îl accesaţi linie cu linie. Pentru a citi o linie dintr-un fişier deschis, puteţi folosi funcţia `fgets()`, care primeşte drept argument o valoare de tip fişier returnată de `fopen()`. De asemenea, trebuie să îi pasaţi funcţiei `fgets()` un al doilea argument de tip număr întreg. Acesta reprezintă numărul de octeţi pe care funcţia trebuie să îi citească dacă nu întâlneşte, mai întâi, sfârşitul de liniei sau de sfârşitul fişierului. Funcţia `fgets()` citeşte din fişier până când ajunge la un caracter de linie nouă (`\n`), la numărul de octeţi specificat în argumentul de lungime sau la sfârşitul fişierului.

```
$line = fgets($fp, 1024); //unde $fp este valoarea de tip fisier
                          //returnata de fopen()
```

Deşi puteţi citi linii cu `fgets()`, trebuie să aveţi o modalitate de a afla când aţi ajuns la sfârşitul fişierului. Funcţia `feof()` vă ajută în această privinţă returnând true atunci când s-a ajuns la sfârşitul fişierului şi false în caz contrar. `feof()` are nevoie de un argument de tip fişier.

```
feof($fp); //unde $fp este valoarea de tip fisier
           //returnata de fopen()
```

Acum aveţi suficiente cunoştinţe pentru a citi un fişier linie cu linie, aşa cum se vede în listingul 11.9

LISTINGUL 11.9. Deschiderea şi citirea unui fişier linie cu linie.

```
 1: <html>
 2: <head>
 3: <title>Listing 11.9 Opening and reading a file line by line</title>
 4: </head>
 5: <body>
 6: <?php
 7: $filename = "test.txt";
 8: $fp = fopen($filename, "r") or die("Couldn't open $filename");
 9: while (!feof($fp)) {
10:     $line = fgets($fp, 1024);
11:     echo "$line<br>";
12: }
13: ?>
14: </body>
15: </html>
```

Dacă acest cod ar fi salvat în directorul rădăcină al serverului dumneavoastră Web şi rulat cu ajutorul browserului, rezultatul ar arăta ca în figura 11.3 (conţinutul fişierului text ar putea fi diferit de cel din exemplu).

Apelăm funcţia `fopen()` pe linia 8 cu numele fişierului pe care dorim să îl citim, folosind operatorul `or` pentru a ne asigura că execuţia scriptului se termină în cazul în care fişierul nu poate fi citit. Acest lucru se întâmplă, de obicei, dacă fişierul nu poate fi citit sau, pe un sistem Unix/Linux, dacă privilegiile de acces ale fişierului

nu îi permit scriptului să acceseze fişierul pentru citire. Citirea efectivă are loc în cadrul buclei instrucţiunii `while` de pe linia 9. Expresia condiţională verificată de instrucţiunea `while` apelează `feof()` la fiecare iteraţie, bucla fiind întreruptă atunci când această funcţie returnează valoarea `true`. Cu alte cuvinte, bucla continuă atâta vreme cât nu se atinge sfârşitul fişierului. În interiorul blocului de cod, folosim `fgets()` pe linia 10 ca să extragem o linie (sau 1024 de octeţi) din fişier. Atribuim rezultatul variabilei `$line` şi o afişăm pe aceasta în browser pe linia 11, adăugându-i un marcaj `<br>` pentru a spori lizibilitatea.

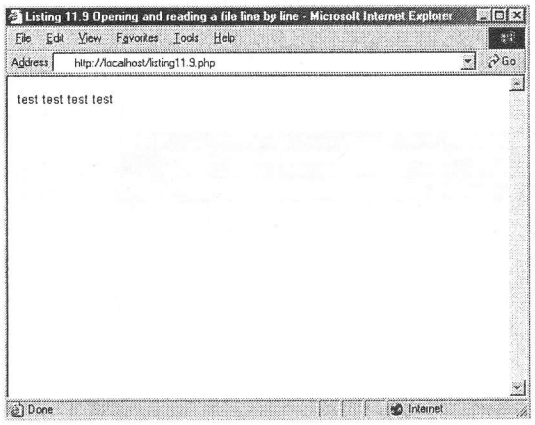


FIGURA 11.3. Rezultatul listingului 11.9.

Citirea unui volum arbitrar de date dintr-un fişier cu ajutorul funcţiei `fread()`

În loc să citiţi textul linie cu linie, puteţi să alegeţi să citiţi bucăţi de dimensiuni arbitrare din fişier. Funcţia `fread()` acceptă drept argument o valoare de tip fişier, precum şi numărul de octeţi pe care doriţi să îl citiţi. `fread()` returnează volumul de date solicitat, cu excepţia cazului în care întâlneşte mai întâi sfârşitul fişierului.

```
$chunk = fread($fp, 16);
```

Listingul 11.10 modifică exemplul anterior astfel încât scriptul să citească din fişier bucăţi de date de 16 octeţi în loc să citească linie cu linie.

LISTINGUL 11.10. Citirea unui fişier cu `fread()`.

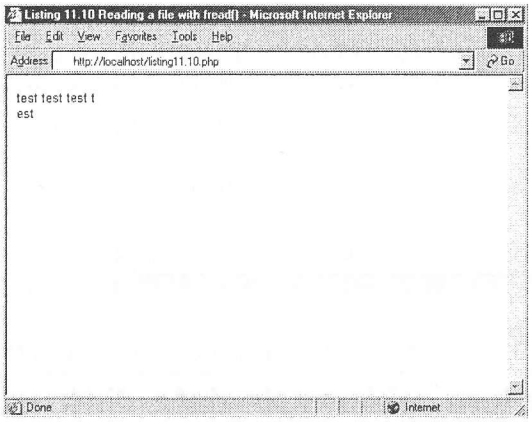
```
1: <html>
2: <head>
3: <title>Listing 11.10 Reading a file with fread()</title>
4: </head>
5: <body>
6: <?php
7: $filename = "test.txt";
```

LISTINGUL 11.10. *(continuare)*

```
 8: $fp = fopen($filename, "r") or die("Couldn't open $filename");
 9: while (!feof($fp)) {
10:     $chunk = fread($fp, 16);
11:     echo "$chunk<br>";
12: }
13: ?>
14: </body>
15: </html>
```

Dacă acest cod ar fi salvat în directorul rădăcină al serverului dumneavoastră Web şi rulat cu ajutorul browserului, rezultatul ar arăta ca în figura 11.4.

FIGURA 11.4. Rezultatul listingului 11.10.



Deşi `fread()` vă permite să definiţi volumul de date ce trebuie preluat din fişier, nu vă lasă să decideţi poziţia de la care să începapă citirea. Puteţi să stabiliţi această poziţie manual, cu ajutorul funcţiei `fseek()`. Aceasta vă permite să schimbaţi poziţia curentă într-un fişier. Ea are nevoie de o valoare de tip fişier şi de un număr întreg care reprezintă decalajul faţă de începutul fişierului (în octeţi) la care vreţi să săriţi.

Listingul 11.11 foloseşte funcţia `fseek()` şi `fread()` pentru a afişa cea de-a doua jumătate a fişierului în browser.

LISTINGUL 11.11. Plimbarea prin fişier cu ajutorul funcţiei `fseek()`.

```
1: <html>
2: <head>
3: <title>Listing 11.11 Moving around a file with fseek()</title>
4: </head>
```

LISTINGUL 11.11. *(continuare)*

```
 5: <body>
 6: <?php
 7: $filename = "test.txt";
 8: $fp = fopen($filename, "r") or die("Couldn't open $filename");
 9: $fsize = filesize($filename);
10: $halfway = (int)($fsize / 2);
11: echo "Halfway point: $halfway <BR>\n";
12: fseek($fp, $halfway);
13: $chunk = fread($fp, ($fsize - $halfway));
14: echo $chunk;
15: ?>
16: </body>
17: </html>
```

Dacă acest cod ar fi salvat în directorul rădăcină al serverului dumneavoastră Web şi rulat cu ajutorul browserului, rezultatul ar arăta ca în figura 11.5.

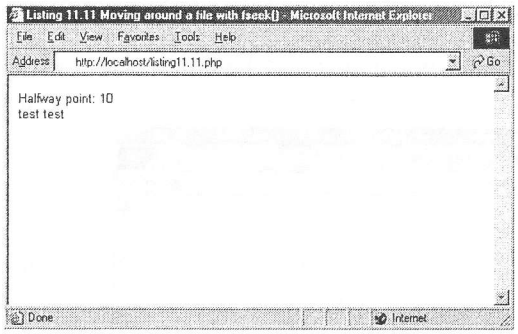


FIGURA 11.5. Rezultatul listingului 11.10.

Calculăm punctul de la jumătatea fişierului împărţind valoarea returnată de funcţia `filesize()` la 2, pe linia 10. Folosim această valoare pasând-o ca al doilea argument funcţiei `fseek()` pe linia 12, sărind astfel la jumătatea fişierului. În fine, apelăm `fread()` pe linia 13 pentru a extrage a doua jumătate a fişierului şi, apoi, afişăm rezultatul în browser.

Citirea de caractere dintr-un fişier cu ajutorul funcţiei `fgetc()`

Funcţia `fgetc()` este similară cu `fgets()`, cu excepţia faptului că returnează numai un singur caracter dintr-un fişier atunci când este apelată. Deoarece un caracter are întotdeauna un octet, `fgetc()` nu are nevoie de un argument de lungime. Trebuie pur şi simplu să îi pasaţi o valoare de tip fişier:

```
$char = fgetc($fp);
```

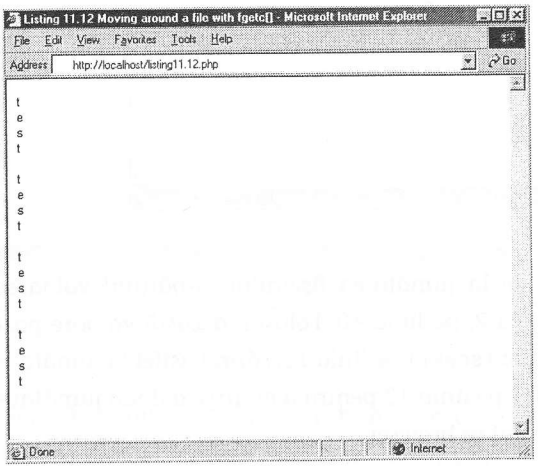
Listingul 11.12 creează o structură repetitivă ce citeşte fişierul `test.txt` caracter cu caracter, afişându-le în browser fiecare pe câte o linie.

LISITNGUL 11.12. Parcurgerea unui fişier cu `fgetc()`.

```
 1: <html>
 2: <head>
 3: <title>Listing 11.12 Moving around a file with fgetc()</title>
 4: </head>
 5: <body>
 6: <?php
 7: $filename = "test.txt";
 8: $fp = fopen($filename, "r") or die("Couldn't open $filename");
 9: while (!feof($fp)) {
10:    $char = fgetc($fp);
11:    echo "$char<BR>";
12: }
13: ?>
14: </body>
15: </html>
```

Dacă acest cod ar fi salvat în directorul rădăcină al serverului dumneavoastră Web şi rulat cu ajutorul browserului, rezultatul ar arăta ca în figura 11.6.

FIGURA 11.6. Rezultatul listingului 11.12.



Scrierea sau adăugarea într-un fişier

Procesul de scriere şi acela de adăugare într-un fişier sunt similare. Diferenţa constă în modul în care apelăm funcţia `fopen()`. Atunci când scrieţi într-un fişier, folosiţi argumentul de mod `"w"` la apelarea funcţiei `fopen()`:

```
$fp = fopen("test.txt", "w");
```

Toate operaţiunile ulterioare de scriere se vor efectua la începutul fişierului. Dacă fişierul nu există deja, el este creat. Dacă fişierul există, conţinutul său este distrus şi înlocuit de datele pe care le scrieţi.

Atunci când adăugaţi într-un fişier, folosiţi argumentul de mod `"a"` în apelul către `fopen()`:

```
$fp = fopen("test.txt", "a");
```

Orice operaţiuni de scriere ulterioare vor adăuga date conţinutului existent, dar, dacă încercaţi să adăugaţi conţinut unui fişier inexistent, fişierul este mai întâi creat.

Scrierea într-un fişier cu ajutorul funcţiei `fwrite()` sau `fputs()`

Funcţia `fwrite()` acceptă drept argumente o valoare de tip fişier şi un şir de caractere şi, apoi, scrie şirul de caractere în fişier. Funcţia `fputs()` funcţionează în exact acelaşi fel:

```
fwrite($fp, "hello world");
fputs($fp, "hello world");
```

Este chiar atât de simplu să scrieţi în fişiere! În listingul 11.13 folosim funcţia `fwrite()` pentru a scrie într-un fişier. După aceea, adăugăm încă un şir de caractere folosind `fputs()`.

LISTINGUL 11.13. Scrierea şi adăugarea într-un fişier.

```
 1: <html>
 2: <head>
 3: <title>Listing 10.13 Writing and appending to a file</title>
 4: </head>
 5: <body>
 6: <?php
 7: $filename = "test.txt";
 8: echo "<p>Writing to $filename ... </p>";
 9: $fp = fopen($filename, "w") or die("Couldn't open $filename");
10: fwrite($fp, "Hello world\n");
11: fclose($fp);
12: echo "<p>Appending to $filename ...</p>";
13: $fp = fopen($filename, "a") or die("Couldn't open $filename");
14: fputs($fp, "And another thing\n");
15: fclose($fp);
16: ?>
17: </body>
18: </html>
```

Rezultatul afişat în browser de acest script este:

```
Writing to test.txt ...

Appending to test.txt ...
```

Dacă deschideţi fişierul `test.txt` sau folosiţi listingul `listing11.9.php` pentru a-i citi conţinutul, veţi descoperi că, acum, fişierul conţine:

```
Hello world
And another thing
```

Blocarea fişierelor cu ajutorul funcţiei `flock()`

Tehnicile de citire şi modificare a fişierelor pe care le-aţi învăţat funcţionează foarte bine dacă scriptul dumneavoastră este folosit de un singur utilizator. În viaţa reală, însă, vă veţi aştepta ca mai mulţi utilizatori să acceseze proiectele dumneavoastră mai mult sau mai puţin în acelaşi timp. Imaginaţi-vă ce s-ar întâmpla dacă doi utilizatori ar executa un script care scrie într-un fişier în acelaşi moment. Fişierul ar deveni corupt imediat.

Pentru prevenirea unei astfel de situaţii, PHP ne pune la dispoziţie funcţia `flock()`. Această funcţie blochează un fişier astfel încât alte procese să fie avertizate să nu scrie sau să citească un fişier în timp ce procesul curent lucrează cu el. Funcţia `flock()` trebuie să primească drept argument o valoare de tip fişier validă şi un număr întreg reprezentând tipul de blocaj pe care doriţi să îl aplicaţi. PHP are implementate constante predefinite pentru fiecare dintre numerele de care veţi avea nevoie. Tabelul 11.1 enumeră trei feluri de blocaje pe care le puteţi aplica unui fişier.

TABELUL 11.1. Argumentele de tip număr întreg ale funcţiei `flock()`.

Constantă	Număr	Tipul blocajului	Descriere
LOCK_SH	1	Partajat	Permite altor procese să citească fişierul dar previne scrierea (se foloseşte atunci când se citeşte un fişier).
LOCK_EX	2	Exclusiv	Interzice oricărui proces atât scrierea cât şi citirea din fişier (se foloseşte atunci când se scrie într-un fişier).
LOCK_UN	3	Eliberare	Eliberează un fişier blocat în modul partajat sau exclusiv.

Ar trebui să apelaţi `flock()` imediat după ce apelaţi `fopen()` şi să o apelaţi din nou pentru a elibera fişierul înainte de îl închide. Dacă blocajul nu este eliberat, nu veţi mai putea citi sau scrie în fişier.

```
$fp = fopen("test.txt", "a") or die("couldn't open");
flock($fp, LOCK_EX); // blocaj exclusiv
// scriere in fisier
flock($fp, LOCK_UN); // eliberarea blocajului
fclose($fp);
```

Pentru mai multe informaţii despre blocarea fişierelor, consultaţi intrarea din Manualul de PHP corespunzătoare funcţiei `flock()` la adresa `http://www.php.net/flock`.

Ştiaţi că...

Manipularea directoarelor

Acum că puteţi verifica, citi şi scrie în fişiere, haideţi să ne concentrăm asupra directoarelor. PHP are numeroase funcţii de manipulare a directoarelor. Haideţi să vedem cum putem să le creăm, şterge şi citi.

Crearea directoarelor cu funcţia `mkdir()`

Funcţia `mkdir()` vă permite să creaţi un director. Funcţia `mkdir()` trebuie să primească drept argumente un şir de caractere reprezentând calea către directorul pe care doriţi să îl creaţi şi un număr întreg octal (în baza 8) care reprezintă modul pe care vreţi să îl atribuiţi directorului nou creat. Veţi specifica un număr octal cu un `0` nesemnificativ. Argumentul de mod are efect numai pe sistemele Unix. Modul trebuie să fie compus din trei cifre între `0` şi `7`, reprezentând permisiunile pentru proprietarul directorului, pentru grupul de utilizatori din care acesta face parte şi, respectiv, pentru restul utilizatorilor. Funcţia `mkdir()` returnează `true` dacă a creat cu succes directorul sau `false` în caz contrar. Dacă `mkdir()` nu reuşeşte, acest lucru este cauzat, de cele mai multe ori, de faptul că directorul conţinător are permisiuni de acces care nu permit proceselor care au ID-ul de utilizator pe care îl are scriptul să scrie în interiorul său. Chiar dacă nu vă este uşor să lucraţi cu permisiunile directoarelor Unix, veţi descoperi că, de cele mai multe ori, veţi avea nevoie de unul dintre exemplele de mai jos. În afara cazului în care doriţi ca directorul dumneavoastră să fie accesibil la scriere tuturor, trebuie să folosiţi, cel mai probabil, `0755`, care permite tuturor să citească directorul dumneavoastră, dar nu şi să scrie în interiorul său.

```
mkdir("testdir", 0777);   // permisiuni globale de scriere / citire
mkdir("testdir", 0755);   // toata lumea si grupul proprietarului: doar
                          // citire / executare
                          // proprietarul: citire / scriere / executare
```

Ştergerea unui director cu ajutorul funcţiei `rmdir()`

Funcţia `rmdir()` vă permite să ştergeţi un director din sistemul de fişiere dacă procesul care rulează scriptul are dreptul să facă acest lucru şi dacă directorul este gol. Funcţia `rmdir()` are nevoie de un singur argument — un şir de caractere reprezentând calea către directorul pe care doriţi să îl creaţi.

```
rmdir("testdir");
```

Deschiderea unui director pentru citire cu `opendir()`

Înainte de a putea citi conţinutul unui director, trebuie să obţineţi o valoare de tip director cu care să operaţi. În acest scop, puteţi folosi funcţia `opendir()`. Acesteia trebuie să îi pasaţi drept argument un şir de caractere reprezentând calea către directorul pe care vreţi să îl deschideţi. Funcţia `opendir()` returnează o valoare tip manipulator (`handle`) de directoare cu excepţia cazurilor în care directorul respectiv nu există sau nu poate fi citit; în aceste cazuri, valoarea returnată va fi `false`.

```
$dh = opendir("testdir");
```

Citirea conţinutului unui director cu ajutorul funcţiei `readdir()`

În acelaşi fel în care folosiţi `fgets()` pentru a citi o linie dintr-un fişier, puteţi folosi `readdir()` pentru a citi câte un nume de fişier sau de director dintr-un director. Funcţia `readdir()` trebuie să primească drept argument un manipulator de directoare şi returnează un şir de caractere conţinând numele elementului cules. Dacă se ajunge la sfârşitul directorului, `readdir()` returnează valoarea false. Reţineţi că `readdir()` returnează doar numele elementelor, nu calea completă a acestora. Listingul 11.14 afişează conţinutul unui director.

LISTINGUL 11.14. Listarea conţinutului unui director cu `readdir()`.

```
 1: <html>
 2: <head>
 3: <title>Listing 11.14 Listing the contents
 4: of a directory with readdir()</title>
 5: </head>
 6: <body>
 7: <?php
 8: $dirname = ".";
 9: $dh = opendir($dirname) or die("couldn't open directory");
10:
11: while (!(($file = readdir($dh)) === false ) ) {
12:     if (is_dir("$dirname/$file")) {
13:             echo "(D) ";
14:     }
15:     echo "$file<br>";
16: }
17: closedir($dh);
18: ?>
19: </body>
20: </html>
```

Dacă acest cod ar fi salvat în directorul rădăcină al serverului dumneavoastră Web şi rulat cu ajutorul browserului, rezultatul ar arăta ca în figura 11.7.

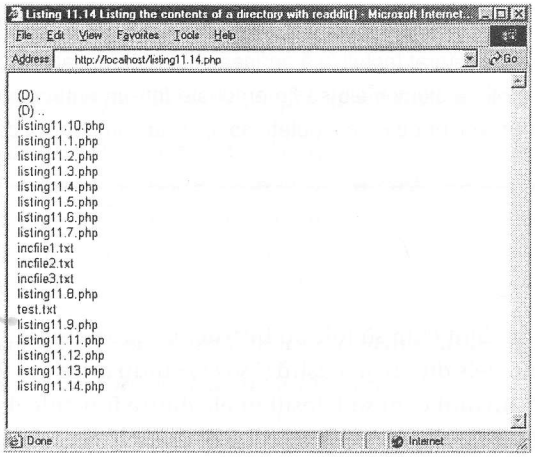


FIGURA 11.7. Rezultatul listingului 11.14.

Deschidem directorul nostru pentru citire apelând funcţia `opendir()` pe linia 9 şi folosim o instrucţiune `while` pentru a parcurge iterativ fiecare dintre elementele acestuia pe linia 11. Apelăm `readdir()` în cadrul expresiei condiţionale a instrucţiunii `while` şi atribuim valoarea returnată variabilei `$file`. În blocul de cod al instrucţiunii `while` folosim variabila `$dirname` în conjuncţie cu variabile `$file` pentru a crea calea completă a elementului, pe care o putem verifica, apoi, pe linia 12. Dacă respectiva cale reprezintă un director, afişăm `(D)` în browser, pe linia 13. În final, afişăm numele elementului pe linia 15.

Am folosit o expresie condiţională precaută în cadrul instrucţiunii `while`. Majoritatea programatorilor de PHP (inclusiv eu) ar fi folosit ceva de genul:

```
while ($file = readdir($dh)) {
      echo "$file<BR>\n";
}
```

Valoarea returnată de `readdir()` este, însă, verificată. Deoarece orice şir diferit de `"Ø"` are valoarea true, n-ar trebui să fie nici o problemă. Imaginaţi-vă, totuşi, un director care conţine patru fişiere: `Ø`, `1`, `2` şi `3`. Pe sistemul meu, rezultatul fragmentului de cod de mai sus ar fi:

```
.
..
```

Atunci când bucla ajunge la fişierul cu numele `Ø`, şirul returnat de `readdir()` are valoarea `false`, fapt ce cauzează întreruperea buclei. Abordarea din listingul 10.14 foloseşte expresia de identitate `===` pentru a verifica dacă valoarea returnată de `readdir()` este echivalentul exact al valorii booleene `false`. Într-un test obişnuit `Ø` este doar asimilat cu `false` şi, astfel, evităm problema.

Apropo

Dacă vi se pare că ordinea elementelor din listarea conținutului unui director este arbitrară, acest lucru este datorat faptului că ordinea este determinată de către sistemul de fișiere. Dacă vreți ca elementele să fie ordonate într-un anumit mod, trebuie să citiți conținutul într-un vector pe care îl puteți, apoi, sorta cum vreți și care poate fi afișat ulterior.

Rezumat

În acest capitol ați învățat cum să folosiți instrucțiunea `include()` ca să încorporați fișiere în documentele dumneavoastră și să executați orice cod PHP conținut în fișierele incluse. Ați învățat cum să folosiți unele dintre funcțiile de verificare a fișierelor disponibile în PHP. Ați explorat funcțiile de citire a fișierelor linie cu linie, caracter cu caracter și în bucăți de dimensiuni arbitrare. Ați învățat să scrieți în fișiere, fie înlocuind sau continuând conținutul deja existent al acestora. La final, ați învățat cum să creați, ștergeți și să citiți directoarele.

Acum că putem să lucrăm cu fișiere, putem salva și accesa volume substanțiale de date. Dacă trebuie să căutăm date în fișiere mari, totuși, scripturile noastre încep să se încetinească destul de mult. Din acest motiv, avem nevoie de o formă sau alta de bază de date.

Întrebări şi răspunsuri

Î. ***Îmi încetineşte execuţia scripturilor folosirea instrucţiunii*** `include()`***?***

R. Deoarece orice fişier inclus trebuie deschis şi procesat de către motorul de PHP, acest lucru adaugă un oarecare cost în termeni de performanţă. Cu toate acestea, beneficiile bibliotecilor de cod reutilizabile sunt, de obicei, mai importante decât aceste costuri în termeni de performanţă, care sunt relativ mici.

Î. ***Ar trebui să opresc întotdeauna execuţia unui script dacă un fişier nu poate fi deschis pentru citire sau scriere?***

R. Ar trebui să prevedeţi întotdeauna această posibilitate. Dacă scriptul are nevoie neapărat de fişierul cu care vreţi să lucraţi, atunci ar trebui să folosiţi `die()`, afişând un mesaj de eroare în browser. În situaţii mai puţin critice, trebuie, totuşi, să prevedeţi eventualitatea aceasta, adăugând, de exemplu, o intrare despre eroare într-un fişier jurnal.

Atelier

Atelierul este conceput să anticipeze posibile întrebări, să vă ajute să recapitulaţi cele învăţate şi să începeţi să vă puneţi cunoştinţele în practică.

Test

1. Ce funcţii aţi folosi pentru a adăuga biblioteci de cod scriptului curent?
2. Ce funcţie folosiţi pentru a afla dacă un fişier există sau nu?
3. Cum determinaţi dimensiunea unui fişier?
4. Ce funcţie folosiţi ca să deschideţi un fişier pentru citire sau scriere?
5. Ce funcţie folosiţi pentru a citi o linie de date dintr-un fişier?
6. Cum puteţi să aflaţi dacă aţi ajuns la sfârşitul unui fişier?
7. Ce funcţie folosiţi pentru a scrie o linie de date într-un fişier?
8. Cum deschideţi un fişier pentru citire?
9. Ce funcţie folosiţi pentru a citi numele unui element dintr-un director după ce aţi deschis un director pentru citire?

Soluții

1. Puteți folosi instrucțiunile `require()` sau `include()` pentru a încorpora fișiere PHP în documentul curent. Puteți să folosiți, de asemenea, instrucțiunea `include_once()` sau `require_once()`.
2. Puteți verifica existența unui fișier folosind funcția `file_exists()`.
3. Funcția `filesize()` returnează dimensiunea în octeți a unui fișier.
4. Funcția `fopen()` deschide un fișier. Ea acceptă calea către un fișier și un caracter reprezentând modul de deschidere. Valoarea returnată este un manipulator de fișiere.
5. Funcția `fgets()` citește datele până la umplerea spațiului pe care i-l indicați, până la sfârșitul liniei, sau până la sfârșitul documentului, oricare este întâlnită prima.
6. Funcția `feof()` returnează true atunci când este întâlnit sfârșitul fișierului al cărui manipulator îi este pasat.
7. Puteți scrie date într-un fișier cu ajutorul funcției `fputs()`.
8. Funcția `opendir()` vă permite să deschideți un fișier pentru citire.
9. Funcția `readdir()` returnează numele unui element al directorului care este deschis.

Activități

1. Creați un formular care să primească numele de familie și prenumele utilizatorului. Scrieți un script care să salveze aceste date într-un fișier.
2. Creați un script care să citească fișierul generat în activitatea 1. În afara afișării conținutului acestui fișier în browser (adăugați un tag `<BR>` la fiecare linie), afișați un rezumat care să includă numărul de linii din fișier și dimensiunea fișierului.

CAPITOLUL 12
Lucrul cu mediul de operare al serverului

PHP, Apache şi MySQL nu sunt singurele aplicaţii de pe calculatorul dumneavoastră. În acest capitol veţi învăţa câteva tehnici de rulare a unora dintre celelalte programe prin intermediul scripturilor dumneavoastră PHP. Deşi exemplele din acest capitol se potrivesc cel mai bine pentru serverele Unix/Linux, aceleaşi principii se aplică şi în cazul utilizatorilor de Windows.

În acest capitol, veţi învăţa cum să:

- transferaţi fluxuri de date înspre şi dinspre aplicaţiile externe;
- trimiteţi comenzi de shell şi să afişaţi rezultatele în browser.

Deschiderea de canale de comunicare (pipes) către şi dinspre procese folosind funcţia `popen()`

În capitolul 11, intitulat *Manipularea fişierelor şi a directoarelor*, aţi învăţat cum să deschideţi un fişier pentru citire şi scriere folosind funcţia `fopen()`. Acum veţi vedea că puteţi să deschideţi un canal de comunicare (pipe) cu un proces cu ajutorul funcţiei `popen()`.

Funcţia `popen()` se are următoarea sintaxă:

```
$pointer_la_fisier = popen("o comanda", mod)
```

Modul poate fi `r` (citire) sau `w` (scriere).

Listingul 12.1 este conceput să provoace o eroare — va încerca să deschidă pentru citire un fişier care nu există.

LISTINGUL 12.1. Folosirea funcţiei `popen()` pentru citirea unui fişier.

```
1: <?php
2: $file_handle = popen("/path/to/fakefile 2>&1", "r");
3: $read = fread($file_handle, 2096);
4: echo $read;
5: pclose($file_handle);
6: ?>
```

Pe linia 2, folosim funcţia `popen()` în încercarea de a deschide un fişier pentru citire. Pe linia 3, este citit eventualul mesaj de eroare stocat în pointerul `$file_handle` şi este afişat pe ecran pe linia 4. În final, pe linia 5 este închis fişierul deschis în linia 2.

Dacă salvaţi acest cod într-un fişier denumit `listing12.1.php`, amplasaţi acest fişier în directorul rădăcină al serverului dumneavoastră Web şi îl accesaţi în browser, veţi vedea afişat pe ecran următorul mesaj de eroare:

```
sh: /path/to/fakefile: No such file or directory (nu există un asemenea fişier
sau director)
```

LISTINGUL 12.2. Folosirea funcţiei `popen()` pentru a citi datele de ieşire ale comenzii Unix `who`.

```
1:  <?php
2:  $handle = popen("who", "r");
3:  while (!feof($handle)) {
4:    $line = fgets($handle,1024);
5:    if (strlen($line) >= 1) {
6:        echo "$line <br>";
7:    }
8:  }
9:  pclose($handle);
10: ?>
```

Pe linia 2, când folosim funcţia `popen()` pentru citire este returnat un pointer la un fişier. Linia 3 începe o buclă `while` care va citi fiecare linie a datelor de ieşire ale procesului şi, în final, va afişa respectiva linie — dacă aceasta conţine informaţii — pe linia 6. Conexiunea este închisă pe linia 9.

Dacă salvaţi acest cod într-un fişier denumit `listing12.2.php`, amplasaţi acest fişier în directorul rădăcină al serverului dumneavoastră Web şi îl accesaţi în browser, veţi vedea ceva asemănător exemplului de mai jos (dar cu informaţiile dumneavoastră, bineînţeles):

```
julie pts/0 Sep 30 21:06 (adsl-67-125-85-212.dsl.snfc21.pacbell.net)
```

Listingul 12.3 vă arată cum să folosiţi funcţia `popen()` în modul de scriere pentru a transfera date unei aplicaţii externe, în cazul acesta column. Scopul scriptului este să preia elementele unui vector multidimensional şi să le scrie sub forma unui tabel într-un fişier ASCII.

LISTINGUL 12.3. Folosirea funcţiei `popen()` pentru transferul de date către comanda Unix `column`.

```
1:  <?php
2:  $products = array(
3:                  array("HAL 2000", 2, "red"),
4:                  array("Tricorder", 3, "blue"),
5:                  array("ORAC AI", 1, "pink"),
6:                  array("Sonic Screwdriver", 1, "orange")
```

LISTINGUL 12.3. *(continuare)*

```
7:              );
8:
9:  $handle = popen("column -tc 3 -s / > /somepath/purchases.txt", "w");
10: foreach ($products as $p) {
11:     fputs($handle, join('/',$p)."\n");
12: }
13: pclose($handle);
14: echo "done";
15: ?>
```

Pe liniile 2–7, este definit un vector multidimensional denumit `$products`. Pe linia 9, `popen()` este folosită în modul de scriere pentru a trimite o comandă către aplicaţia `column`. Comanda îi trimite aplicaţiei `column` parametri prin care îi spune să formateze datele de intrare sub forma unui tabel, folosind caracterul / ca delimitator de câmp. Rezultatul va fi trimis către un fişier denumit `purchases.txt`. Liniile 10–12 folosesc instrucţiunea `foreach` ca să parcurgă iterativ elementele vectorului `$products` şi să trimită fiecare element către pointerul fişierului deschis. Funcţia `join()` este folosită pentru a converti vectorul într-un şir de caractere căruia i se adaugă şi caracterul de delimitare. Linia 13 închide pointerul fişierului deschis, iar linia 14 afişează starea pe ecran.

Dacă salvaţi acest cod într-un fişier denumit `listing12.3.php`, amplasaţi acest fişier în directorul rădăcină al serverului dumneavoastră Web şi îl accesaţi în browser, scriptul ar trebui să creeze un fişier în calea specificată. Dacă mă uit la fişierul creat pe calculatorul meu, văd următoarele:

```
HAL 2000          2  red
Tricorder         3  blue
ORAC AI           1  pink
Sonic Screwdriver 1  orange
```

Executarea de comenzi cu ajutorul funcţiei `exec()`

Funcţia `exec()` este una dintre cele câteva funcţii pe care le puteţi folosi pentru a transmite comenzi către interpretorul de comenzi al sistemului (shell). Funcţia `exec()` are nevoie de un argument de tip şir de caractere reprezentând calea către comanda pe care doriţi să o executaţi şi acceptă, opţional, o variabilă de tip vector în care vor fi stocate datele de ieşire ale comenzii, precum şi o variabilă scalară ce va conţine valoarea returnată de comandă (1 sau 0). De exemplu:

```
exec("/path/to/somecommand", $output_array, $return_val);
```

Listingul 12.4 foloseşte funcţia `exec()` pentru a lista directoarele cu ajutorul comenzii de shell `ls`.

LISTINGUL 12.4. Folosirea funcției `exec()` și a comenzii `ls` pentru a enumera directoarele.

```
1: <?php
2: exec("ls -al .", $output_array, $return_val);
3: echo "Returned $return_val<br>";
4: foreach ($output_array as $o) {
5:     echo "$o <br>";
6: }
7: ?>
```

Pe linia 2, funcția `exec()` trimite comanda `ls` astfel încât datele de ieșire ale comenzii să fie stocate în vectorul `$output_array`, iar valoarea returnată de comandă în variabila `$return_val`. Linia 3 nu face decât să afișeze valoarea returnată, în timp ce bucla `foreach` din liniile 4–6 afișează fiecare element al vectorului `$output_array`.

Dacă salvați acest cod într-un fișier denumit `listing12.4.php`, amplasați acest fișier în directorul rădăcină al serverului dumneavoastră Web și îl accesați în browser, s-ar putea să vedeți ceva asemănător cu figura 12.1 (cu informațiile dumneavoastră, nu cu ale mele, bineînțeles).

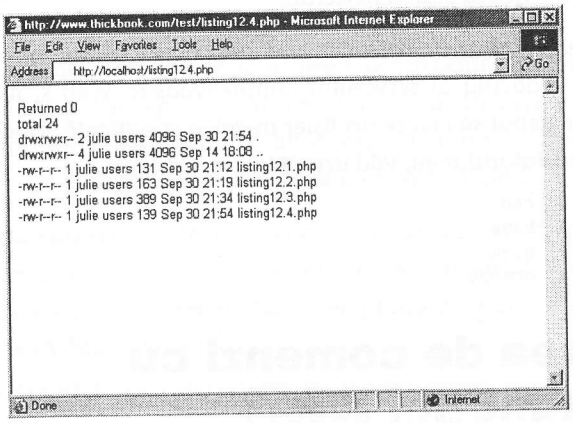


FIGURA 12.1. Rezultatul scriptului care folosește funcția `exec()` pentru a lista conținutul directorului.

În capitolul precedent ați învățat să folosiți funcțiile `opendir()` și `readdir()` pentru a lista conținutul unui director, dar aici am exemplificat într-un mod simplu folosirea funcției `exec()` pentru a efectua aceeași operațiune utilizând instrumente de sistem. Vor exista momente în care executarea unei comenzi de sistem va avea un efect a cărui obținere ar necesita un efort enorm de scriere de cod PHP. De exemplu, s-ar putea să fi creat deja un script de shell sau un script Perl care să efectueze o sarcină complexă într-un timp scurt; în loc să reinventați roata scriind același lucru în PHP, mai bine folosiți funcția `exec()` pentru accesa scriptul deja existent. Cu toate acestea, rețineți că apelurile către procese externe vor reduce performanțele scriptului în termeni de viteză și de consum de memorie.

Executarea comenzilor folosind `system()` sau `passthru()`

Funcţia `system()` este similară cu funcţia `exec()` prin faptul că lansează o aplicaţie externă şi că utilizează o variabilă scalară pentru a stoca valoarea returnată:

```
system("/path/to/somecommand", $return_val);
```

Funcţia `system()` este diferită de funcţia `exec()` prin aceea că trimite datele de ieşire direct către browser, fără intervenţie din partea scriptului. Următoarea bucăţică de cod foloseşte funcţia `system()` pentru a afişa o pagină de manual Unix apelând comanda `man`, formatată cu ajutorul perechii de marcaje HTML `<pre></pre>`:

```
<?php
echo "<pre>";
system("man man | col -b", $return_val);
echo "</pre>";
?>
```

Funcţia `passthru()` are aceeaşi sintaxă ca funcţia `system()`, dar se comportă diferit. Atunci când folosiţi `passthru()`, datele de ieşire ale comenzii de shell sunt trimise înapoi în timp real, fără a mai fi strânse într-o zonă de memorie tampon; acest lucru este util atunci când executăm comenzi care produc date de tip binar, nu text simplu. Un exemplu ar fi folosirea instrumentelor de shell pentru a găsi o imagine şi a o trimite înapoi către browser, după cum se poate vedea în listingul 12.5.

LISTINGUL 12.5. Folosirea funcţiei `passthru()` pentru a afişa date de tip binar.

```
1: <?php
2: if ((isset($_GET[imagename])) && (file_exists($_GET[imagename]))) {
3:    header("Content-type: image/gif");
4:    passthru("giftopnm $_GET[imagename] | pnmscale -xscale .5 -yscale
                .5 | ppmtogif");
5: } else {
6:     echo "The image $_GET[imagename] could not be found";
7: }
8: ?>
```

Apropo

Utilitarele de shell folosite în acest script, `giftopnm`, `pnmscale` şi `ppmtogif` pot fi instalate pe sistemul dumneavoastră sau nu. Dacă nu sunt instalate, nu vă faceţi griji; folosiţi acest listing doar pentru a înţelege ideea de utilizare a funcţiei `passthru()`.

Acest script ar putea fi apelat din HTML în felul următor:

```
<img src="listing12.5.php?imagename=<?php echo urlencode("test.gif") ?>">
```

Pe linia 2 a listingului 12.5, şirul introdus de utilizator este verificat pentru a se asigura că fişierul respectiv (`test.gif`, în linia de cod HTML dată) există. Din cauză că scriptul va trimite datele în format GIF către browser, antetul corespunzător este trimis pe linia 3. Pe linia 4, funcţia `passthru()` execută trei comenzi diferite consecutiv — `giftopnm`, `pnmscale` şi `ppmtogif`, care micşorează imaginea la 50% din înălţimea şi lăţimea originalului. Datele de ieşire ale comenzii `passthru()` — adică datele reprezentând noua imagine — sunt trimise direct către browser.

Apropo

În acest exemplu, precum şi în altele care au de-a face cu sistemul, aţi fi putut să folosiţi funcţiile `escapeshellcmd()` sau `escapeshellarg()` pentru a preveni procesarea elementelor din datele introduse de utilizator. Făcând astfel, vă asiguraţi că un utilizator nu poate să păcălească sistemul să execute comenzi arbitrare cum ar fi ştergerea unor fişiere de sistem importante sau modificarea parolelor. Aceste funcţii sunt aplicate instanţei iniţiale a datelor introduse de utilizator, cum ar fi:

```
$date_nou_introduse = escapeshellcmd($_GET[niste_date_introduse]);
```

Apoi, puteţi să faceţi referire la variabila `$date_nou_introduse` peste tot în restul scriptului în loc de `$_GET[niste_date_introduse]`. Pentru a păstra securitatea sistemului, puteţi, deci, folosi aceste două comenzi, şi trebuie să vă asiguraţi că scriptul este scris în aşa fel încât să nu efectueze decât operaţiunile pe care le doriţi dumneavoastră, şi nu pe cele pe care utilizatorii rău voitori le-ar putea dori.

Rezumat

În acest scurt capitol vi s-au prezentat câteva metode de comunicare cu sistemul de operare şi cu aplicaţiile externe. Deşi PHP este un limbaj extrem de vast şi de robust, se poate să decideţi că este mai eficient din punct de vedere al costurilor şi timpului să utilizaţi scripturile deja existente scrise în alte limbaje precum C sau Perl. Puteţi accesa aceste aplicaţii externe folosind funcţiile `popen()`, `exec()`, `system()` şi `passthru()`.

Aţi învăţat cum să transferaţi date către o comandă folosind `popen()`, lucru foarte util pentru aplicaţiile care acceptă date de la consolă şi atunci când vreţi să procesaţi datele aşa cum vă sunt trimise de către o aplicaţie. Aţi mai învăţat şi cum să folosiţi funcţiile `exec()` şi `system()` ca să transmiteţi comenzi către shell şi să acceptaţi datele de tip binar rezultate din executarea unei comenzi de shell.

Întrebări şi răspunsuri

Î. Unde pot găsi mai multe informaţii despre securitatea pe Web?

A. Una dintre sursele de autoritate care pune la dispoziţie date introductive despre securitatea pe Web este „The World Wide Web Security FAQ" (*Întrebări frecvente despre securitatea WWW*), un document scris de Lincoln Stein, pe care îl puteţi găsi la adresa `http://www.w3.org/Security/Faq/`.

Atelier

Atelierul este conceput să anticipeze posibile întrebări, să vă ajute să recapitulaţi cele învăţate şi să începeţi să vă puneţi cunoştinţele în practică.

Test

1. Ce funcţie se foloseşte pentru deschiderea unui canal de comunicare (pipe) cu un proces?
2. Cum puteţi citi date de la un proces după ce aţi deschis o conexiune? Şi cum puteţi scrie date?
3. Cum puteţi să preveniţi procesarea elementelor din datele introduse de utilizatori pentru a le face un pic mai sigure înainte de a le transmite către o comandă de shell?

Soluţii

1. `popen()`
2. Puteţi citi şi scrie de la şi, respectiv, către un proces exact cum aţi face-o în cazul unui fişier deschis, şi anume cu `feof()` şi `fgets()` pentru citire şi `fputs()` pentru scriere.
3. Puteţi folosi funcţiile `escapeshellcmd()` şi `escapeshellarg()` atunci când trebuie să acceptaţi date de la utilizator.

CAPITOLUL 13
Manipularea imaginilor

O instalare standard de PHP are multe funcţii predefinite pentru crearea dinamică şi manipularea imaginilor. Cu câteva adăugiri, puteţi extinde şi mai mult funcţionalitatea. În acest capitol veţi învăţa lucrurile de bază despre crearea şi manipularea imaginilor cu ajutorul funcţiilor PHP:

- cum să modificaţi instalarea de PHP pentru a spori funcţionalitatea legată de imagini;
- cum să creaţi o imagine nouă;
- cum să modificaţi imagini existente.

Să înţelegem procesul de creare a unei imagini

Crearea unei imagini în PHP nu este asemănătoare creării unei imagini cu ajutorul unui program de desen. Nu putem folosi mouse-ul pentru a desena şi nu există instrumente de umplere a spaţiilor predefinite cu o anumită culoare. Mai mult, nu există posibilitatea de a folosi opţiunile de salvare a fişierului cu care, în programul de desen, pot fi create automat fişiere GIF sau JPEG sau PNG (şi aşa mai departe) prin simpla folosire a meniului Save As (salvează ca).

Din contră, va trebui să deveniţi dumneavoastră aplicaţia de desen. În calitate de programator, trebuie să creaţi fiecare pas scriind cod PHP. Aveţi nevoie să folosiţi funcţii PHP individuale ca să definiţi culori, să desenaţi şi să umpleţi forme, să dimensionaţi şi redimensionaţi imaginea şi să salvaţi imaginea sub forma unui anume tip de fişier. Nu este atât de dificil pe cât pare, totuşi, dacă înţelegeţi fiecare pas necesar şi efectuaţi operaţiunile în ordine.

Câteva cuvinte despre culoare

Atunci când definiţi culori în scripturile dumneavoastră care lucrează cu imagini, va trebui să folosiţi sistemul de culoare RGB. Puteţi defini o culoare anume utilizând valori numere întregi cuprinse între 0 şi 255 pentru fiecare dintre canalele roşu (Red), verde (Green) şi albastru (Blue). Valoarea 0 indică absenţa culorii respective, iar valoarea 255 indică o saturaţie maximă.

De exemplu, valoarea RGB pentru roşul pur este (255, 0, 0) sau alocarea maximă a valorii pentru roşu, fără verde şi fără albastru. În mod similar, verdele pur are valoarea (0, 255, 0), iar albastrul pur are valoarea (0, 0, 255). Albul, pe de altă parte, are valoarea RGB (255, 255, 255), în vreme ce negrul este reprezentat de valoarea (0, 0, 0). O nuanţă drăguţă de purpuriu are valoarea (153, 51, 153) şi o nuanţă deschisă de gri poate fi reprezentată, în sistemul RGB, prin (204, 204, 204).

Modificările ce trebuie făcute instalării de PHP

Începând cu versiunea 4.3.0, distribuţia de PHP cuprinde o versiune a bibliotecii grafice GD scrisă de Thomas Boutell. Includerea acestei biblioteci elimină nevoia descărcării de pe Internet şi instalării diverselor biblioteci create de terţi, dar această bibliotecă trebuie activată la momentul instalării.

Apropo

Dacă nu puteţi instala PHP aşa cum aţi fost instruiţi în capitolul 3, intitulat *Instalarea şi configurarea pachetului PHP*, şi sunteţi nevoit să lucraţi cu o versiune de PHP anterioară versiunii 4.3.0, va trebui să mergeţi la `http://www.boutell.com/gd/` şi să descărcaţi codul sursă al bibliotecii GD. Urmaţi instrucţiunile incluse în acel pachet de software şi consultaţi-i manualul de utilizare dacă aveţi dificultăţi la instalare.

Pentru a activa folosirea bibliotecii GD la momentul instalării, utilizatorii de Linux/Unix trebuie să adauge următorii parametri la comanda configure atunci când se pregătesc să compileze PHP-ul:

```
--with-gd
```

Apropo

Dacă aţi descărcat singur o versiune de GD, trebuie să specificaţi şi calea, cum ar fi:

```
--with-gd=/path/to/gd.
```

După ce aţi rulat programul `configure` din nou, trebuie să parcurgeţi procesul `make` şi `make install` aşa cum aţi făcut în capitolul 3. Utilizatorii de Windows care vor să activeze biblioteca GD trebuie doar să activeze `php_gd2.dll` ca extensie în fişierul `php.ini`, aşa cum s-a explicat în capitolul 3.

Atunci când folosiţi biblioteca GD sunteţi limitat de faptul că nu puteţi lucra decât cu fişiere în format GIF. Totuşi, instalând biblioteci suplimentare, puteţi lucra şi cu fişiere JPEG şi PNG.

Obținerea bibliotecilor suplimentare

Faptul că puteți lucra cu fișiere GIF poate că răspunde perfect nevoilor dumneavoastră, dar, dacă vreți să creați fișiere JPEG sau PNG, trebuie să descărcați și să instalați câteva biblioteci și să faceți câteva modificări instalării dumneavoastră de PHP:

- bibliotecile JPEG și informații despre acestea pot fi găsite la adresa `ftp://ftp.uu.net/graphics/jpeg/`.
- bibliotecile PNG și informații despre acestea sunt disponibilie la adresa `http://www.libpng.org/pub/png/libpng.html`.
- dacă lucrați cu fișiere PNG, ar trebui să instalați și biblioteca `zlib`, pe care o găsiți la `http://www.gzip.org/zlib/`.

Urmați instrucțiunile de pe aceste site-uri pentru a instala respectivele biblioteci. După instalare, utilizatorii de Linux/Unix trebuie să reconfigureze din nou și să recompileze PHP-ul, adăugând mai întâi următoarele la parametrii comenzii `configure` (presupunând că vreți să folosiți toate cele trei biblioteci):

```
--with-jpeg-dir=[calea catre directorul jpeg]
--with-png-dir=[calea catre directorul PNG]
--with-zlib=[calea catre directorul zlib]
```

După ce ați rulat programul `configure` din nou, trebuie să parcurgeți procesul `make` și `make install` așa cum ați făcut în capitolul 3. Bibliotecile dumneavoastră ar trebui să fie acum activate și gata de folosire.

Desenarea unei imagini noi

Funcția PHP de bază ce este folosită pentru crearea unei imagini noi se numește `ImageCreate()`, dar crearea unei imagini nu constă pur și simplu din apelarea acestei funcții. Crearea unei imagini este un proces secvențial și include folosirea mai multor funcții PHP.

Crearea unei imagini începe cu funcția `ImageCreate()`, dar tot ce face această funcție este să rezerve spațiul de memorie pentru suprafața goală pe care veți desena imaginea. Următoarea linie creează o suprafață lată de 150 de pixeli și înaltă de 150 de pixeli:

```
$myImage = ImageCreate(150,150);
```

Având acum suprafața de lucru definită, trebuie să definim câteva culori pe care le vom folosi în noua imagine. Următoarele exemple definesc cinci astfel de culori, folosind funcția `ImageColorAllocate()` și valori RGB:

```
$black = ImageColorAllocate($myImage, 0, 0, 0);
$white = ImageColorAllocate($myImage, 255, 255, 255);
$red   = ImageColorAllocate($myImage, 255, 0, 0);
```

```
$green = ImageColorAllocate($myImage, 0, 255, 0);
$blue = ImageColorAllocate($myImage, 0, 0, 255);
```

Apropo

> În scripturile dumneavoastră, prima culoare pe care o definiți este folosită drept culoare de fundal pentru imagine. În acest caz, fundalul va fi negru.

Acum că ştiţi care sunt elementele de bază ale iniţializării suprafeţei de desen şi ale alocării culorilor, puteţi să începeţi să învăţaţi cum se desenează forme şi cum se afişează efectiv imaginea într-un browser.

Desenarea formelor şi a liniilor

Câteva funcţii PHP vă pot ajuta să desenaţi forme şi linii pe suprafaţa dumneavoastră de desen:

- `ImageEllipse()` este folosită pentru desenarea unei elipse;
- `ImageArc()` este folosită pentru desenarea unei porţiune de elipsă;
- `ImagePolygon()` este folosită pentru desenarea unui poligon;
- `ImageRectangle()` este folosită pentru desenarea unui dreptunghi;
- `ImageLine()` este folosită pentru desenarea unei linii.

Folosirea acestor funcţii necesită un pic de planificare deoarece trebuie să stabiliţi punctele de la care să pornească şi la care să se oprească desenul. Fiecare dintre aceste funcţii foloseşte coordonate pe axele x şi y pentru indicarea locului din care să înceapă desenul pe suprafaţa de desen. Trebuie, de asemenea, să definiţi distanţele pe axele x şi y pe care trebuie să se întindă desenul.

De exemplu, următoarea linie de cod va desena un dreptunghi pe suprafaţa de desen începând de la punctul cu coordonatele `(15,15)` şi întinzându-se 25 de pixeli pe orizontală şi 40 de pixeli pe verticală, astfel încât liniile să se oprească în punctul cu coordonatele `(40,55)`. În plus, liniile vor fi desenate cu culoarea roşie, care a fost deja definită în variabile `$red`.

```
ImageRectangle($myImage, 15, 15, 40, 55, $red);
```

Dacă vreţi să desenaţi un alt dreptunghi de aceeaşi dimensiune dar cu linii albe, începând din punctul în care s-a oprit dreptunghiul anterior, ar trebui să folosiţi următoarea linie de cod:

```
ImageRectangle($myImage, 40, 55, 65, 95, $white);
```

Listingul 13.1 arată scriptul de creare a imaginilor de până acum, cu câteva linii de cod în plus care afişează imaginea într-un browser.

LISTINGUL 13.1. Crearea unei imagini noi.

```
1:  <?
2:  //creeaza suprafata de desenat
3:  $myImage = ImageCreate(150,150);
4:
5:  //defineste cateva culori
6:  $black = ImageColorAllocate($myImage, 0, 0, 0);
7:  $white = ImageColorAllocate($myImage, 255, 255, 255);
8:  $red  = ImageColorAllocate($myImage, 255, 0, 0);
9:  $green = ImageColorAllocate($myImage, 0, 255, 0);
10: $blue = ImageColorAllocate($myImage, 0, 0, 255);
11:
12: //deseneaza cateva dreptunghiuri
13: ImageRectangle($myImage, 15, 15, 40, 55, $red);
14: ImageRectangle($myImage, 40, 55, 65, 95, $white);
15:
16: //afiseaza imaginea in browser
17: header ("Content-type: image/jpeg");
18: ImageJpeg($myImage);
19:
20: //curata suprafata de lucru
21: ImageDestroy($myImage);
21: ?>
```

Liniile 17–18 trimit fluxul de date reprezentând imaginea către browser trimiţând, mai întâi, antetul corespunzător cu ajutorul funcţiei `header()`, folosind tipul MIME al imaginii care este create. Apoi, este folosită, după caz, una dintre funcţiile `ImageGif()`, `ImageJpeg()` sau `ImagePng()` pentru a trimite fluxul de date; acest exemplu afişează o imagine de tip JPEG. Pe linia 21 este utilizată funcţia `ImageDestroy()` pentru eliberarea memoriei folosite de funcţia `ImageCreate()`.

Salvaţi listingul sub numele `imagecreate.php` şi amplasaţi acest fişier în directorul rădăcină al serverului dumneavoastră Web. Atunci când este accesat, în browser trebuie să apară ceva similar cu figura 13.1, dar în culori.

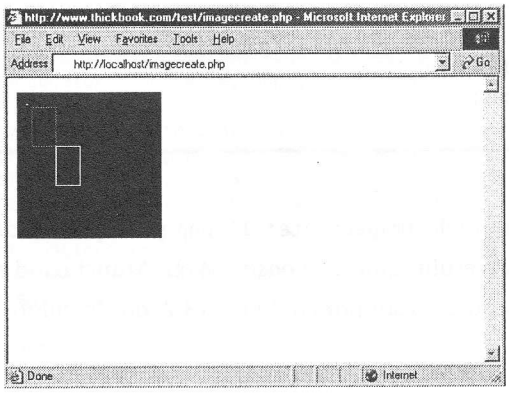


FIGURA 13.1. O suprafaţă de desen pe care au fost desenate două dreptunghiuri.

Umplerea suprafețelor cu culoare

Rezultatul listingului 13.1 a afişat numai conturul dreptunghiurilor. PHP are şi funcţii concepute să umple suprafeţele:

- `ImageFilledEllipse()` se foloseşte pentru a umple o elipsă;
- `ImageFilledArc()` se foloseşte pentru a umple o porţiune de elipsă;
- `ImageFilledPolygon()` se foloseşte pentru a umple un poligon;
- `ImageFilledRectangle()` se foloseşte pentru a umple un dreptunghi.

Aceste funcţii se folosesc la fel ca şi corespondentele lor fără umplere. În listingul 13.2, funcţiile fără umplere au fost înlocuite cu funcţiile concepute să umple o suprafaţă.

LISTINGUL 13.2. Crearea unei noi imagini cu suprafețe colorate pline.

```
1:  <?
2:  //creeaza suprafata de desenat
3:  $myImage = ImageCreate(150,150);
4:
5:  //defineste cateva culori
6:  $black = ImageColorAllocate($myImage, 0, 0, 0);
7:  $white = ImageColorAllocate($myImage, 255, 255, 255);
8:  $red  = ImageColorAllocate($myImage, 255, 0, 0);
9:  $green = ImageColorAllocate($myImage, 0, 255, 0);
10: $blue = ImageColorAllocate($myImage, 0, 0, 255);
11:
12: //deseneaza cateva dreptunghiuri
13: ImageFilledRectangle($myImage, 15, 15, 40, 55, $red);
14: ImageFilledRectangle($myImage, 40, 55, 65, 95, $white);
15:
16: //afiseaza imaginea in browser
17: header ("Content-type: image/jpeg");
18: ImageJpeg($myImage);
19:
20: //curata suprafata de lucru
21: ImageDestroy($myImage);
21: ?>
```

Salvaţi listingul sub numele `imagecreatefill.php` şi amplasaţi acest fişier în directorul rădăcină al serverului dumneavoastră Web. Atunci când este accesat, în browser trebuie să apară ceva similar cu figura 13.2, dar în culori.

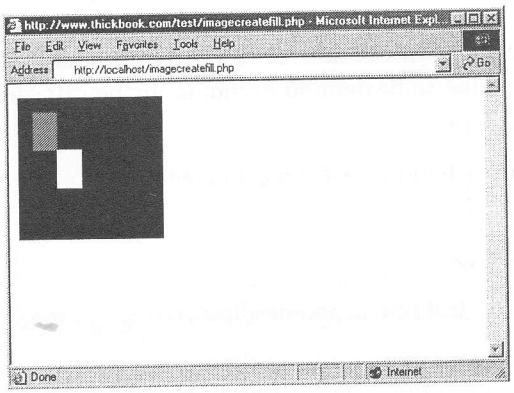


FIGURA 13.2. Suprafața de desen, cu două dreptunghiuri colorate.

Înfrumusețarea paginilor cu ajutorul diagramelor circulare

Exemplele precedente au fost destul de plictisitoare, dar v-au prezentat procesul de creare al imaginilor — definirea suprafeței de desen, definirea culorilor şi, apoi, desenarea contururilor sau a suprafețelor pline. Puteți să folosiți aceeaşi succesiune de acțiuni pentru a permite scripturilor dumneavoastră să creeze diagrame şi grafice pe baza unor date statice sau dinamice. Listingul 13.3 desenează o diagramă circulară simplă.

LISTINGUL 13.3. O diagramă circulară simplă.

```
1:  <?
2:  //creează suprafata de desenat
3:  $myImage = ImageCreate(150,150);
4:
5:  //defineste cateva culori
6:  $white = ImageColorAllocate($myImage, 255, 255, 255);
7:  $red  = ImageColorAllocate($myImage, 255, 0, 0);
8:  $green = ImageColorAllocate($myImage, 0, 255, 0);
9:  $blue = ImageColorAllocate($myImage, 0, 0, 255);
10:
11: //deseneaza un grafic tip placinta
12: ImageFilledArc($myImage, 50, 50, 100, 50, 0, 90, $red, IMG_ARC_PIE);
13: ImageFilledArc($myImage, 50, 50, 100, 50, 91, 180 , $green, IMG_ARC_PIE);
14: ImageFilledArc($myImage, 50, 50, 100, 50, 181, 360 , $blue, IMG_ARC_PIE);
15:
16: //afiseaza imaginea in browser
17: header ("Content-type: image/jpeg");
18: ImageJpeg($myImage);
19:
20: //curata suprafata de lucru
21: ImageDestroy($myImage);
22: ?>
```

Cu excepţia liniilor 12–14, acest script este aproape identic cu cele din listingurile anterioare. Spre deosebire de acelea, aici nu am mai definit culoarea neagră, lăsând culoarea albă să fie definită prima, motiv pentru care aceasta va fi culoarea suprafeţei de desen.

Pe liniile 12–14, funcţia folosită este `ImageFilledArc()`, ce primeşte câteva argumente:

- identificatorul imaginii;
- coordonata x a centrului porţiunii de elipsă;
- coordonata y a centrului porţiunii de elipsă;
- lăţimea porţiunii de elipsă;
- înălţimea porţiunii de elipsă;
- punctul de start al porţiunii de elipsă;
- punctul de sfârşit al porţiunii de elipsă;
- culoarea;
- stilul.

Priviţi cu atenţie linia 14 din listingul 13.3:

```
14: ImageFilledArc($myImage, 50, 50, 100, 50, 181, 360 , $blue, IMG_ARC_PIE);
```

Această linie de cod spune că arcul de elipsă trebuie să înceapă din punctul cu coordonatele (50, 50) şi să aibă o lăţime de 100 şi o înălţime de 50. Punctul de start (în grade de unghi) este 181, iar punctul de sfârşit este 360. Arcul trebuie să fie umplut cu culoare predefinită `$blue` şi trebuie să folosească stilul `IMG_ARC_PIE`. Acesta este unul dintre cele câteva stiluri predefinite care pot fi folosite pentru afişare; `IMG_ARC_PIE` spune că trebuie creată o margine rotunjită. Puteţi să aflaţi mai multe despre diversele stiluri din Manualul de PHP, la adresa: `http://www.php.net/image`.

Salvaţi listingul sub numele `imagecreatepie.php` şi amplasaţi acest fişier în directorul rădăcină al serverului dumneavoastră Web. Atunci când este accesat, în browser trebuie să apară ceva similar cu figura 13.3, dar în culori.

Puteţi extinde codul din listingul 13.3 pentru a-i da graficului dumneavoastră un aspect 3D. Pentru a face acest lucru, trebuie să definiţi încă trei culori pentru margine. Aceste culori pot fi nuanţe mai deschise sau mai închise ale culorilor de bază, astfel încât să permită un grad de contrast. Următoarele exemple definesc nuanţe mai deschise:

```
$lt_red = ImageColorAllocate($myImage, 255, 150, 150);
$lt_green = ImageColorAllocate($myImage, 150, 255, 150);
$lt_blue = ImageColorAllocate($myImage, 150, 150, 255);
```

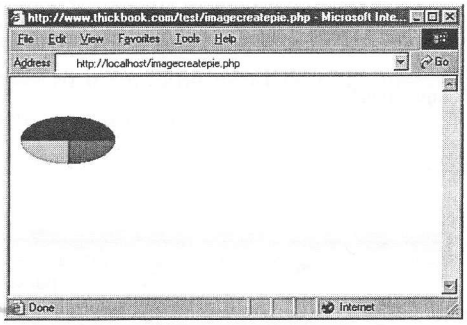


FIGURA 13.3. Un grafic de tip diagramă circulară, cu feliile sale.

Pentru a crea un efect de umbră, puteţi folosi o buclă `for` care să adauge o serie de arcuri mai mici de la punctul (50, 60) până la (50, 51), folosind culori de umplere mai deschise:

```
for ($i = 60;$i > 50;$i--) {
     ImageFilledArc($myImage, 50, $i, 100, 50, 0, 90, $lt_red, IMG_ARC_PIE);
     ImageFilledArc($myImage, 50, $i, 100, 50, 91, 180, $lt_green, IMG_ARC_PIE);
     ImageFilledArc($myImage, 50, $i, 100, 50, 181, 360, $lt_blue, IMG_ARC_PIE);
}
```

Listingul 13.4 vă arată codul folosit pentru crearea unui grafic de tip diagramă circulară 3D.

LISTINGUL 13.4. Un grafic de tip diagramă circulară 3D.

```
1:  <?
2:  //creeaza suprafata de desenat
3:  $myImage = ImageCreate(150,150);
4:
5:  //defineste cateva culori
6:  $white = ImageColorAllocate($myImage, 255, 255, 255);
7:  $red  = ImageColorAllocate($myImage, 255, 0, 0);
8:  $green = ImageColorAllocate($myImage, 0, 255, 0);
9:  $blue = ImageColorAllocate($myImage, 0, 0, 255);
10: $lt_red = ImageColorAllocate($myImage, 255, 150, 150);
11: $lt_green = ImageColorAllocate($myImage, 150, 255, 150);
12: $lt_blue = ImageColorAllocate($myImage, 150, 150, 255);
13:
14: //deseneaza zona de umbra
15: for ($i = 60;$i > 50;$i--) {
16:     ImageFilledArc($myImage, 50, $i, 100, 50, 0, 90, $lt_red, IMG_ARC_PIE);
17:     ImageFilledArc($myImage, 50, $i, 100, 50, 91, 180, $lt_green,
        IMG_ARC_PIE);
18:     ImageFilledArc($myImage, 50, $i, 100, 50, 181, 360, $lt_blue,
        IMG_ARC_PIE);
19: }
20:
21: //deseneaza un grafic tip diagrama circulara
22: ImageFilledArc($myImage, 50, 50, 100, 50, 0, 90, $red, IMG_ARC_PIE);
23: ImageFilledArc($myImage, 50, 50, 100, 50, 91, 180 , $green, IMG_ARC_PIE);
24: ImageFilledArc($myImage, 50, 50, 100, 50, 181, 360 , $blue, IMG_ARC_PIE);
```

LISTINGUL 13.4. (*continuare*)

```
25:
26: //afiseaza imaginea in browser
27: header ("Content-type: image/jpeg");
28: ImageJpeg($myImage);
29:
30: //curata suprafata de lucru
31: ImageDestroy($myImage);
32: ?>
```

Salvaţi listingul sub numele `imagecreate3dpie.php` şi amplasaţi acest fişier în directorul rădăcină al serverului dumneavoastră Web. Atunci când este accesat, în browser trebuie să apară ceva similar cu figura 13.4, dar în culori.

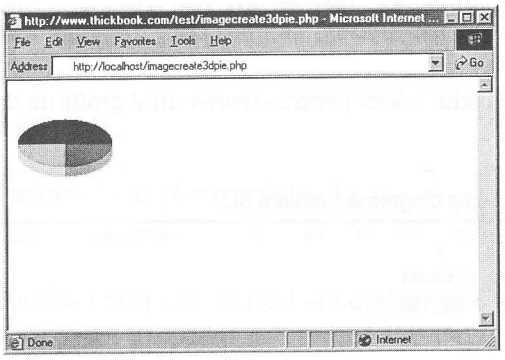


FIGURA 13.4. Un grafic de tip diagramă circulară 3D, cu feliile sale.

Acestea sunt doar nişte exemple foarte simpliste care arată puterea unora dintre funcţiile de desen şi umplere din PHP. În secţiunea următoare, veţi învăţa să manipulaţi imagini deja existente.

Modificarea imaginilor deja existente

Procesul de creare a imaginilor pe baza altor imagini se bazează pe aceiaşi paşi esenţiali ca şi crearea unei imagini noi — diferenţa constă în suprafaţa de desen cu care se lucrează. Adineauri, aţi creat o suprafaţă de desen nouă folosind funcţia `ImageCreate()`. Atunci când vreţi să creaţi o imagine pe baza unei alte imagini, trebuie să folosiţi familia de funcţii `ImageCreateFrom*()`.

Puteţi crea imagini pe baza fişierelor deja existente de tip GIF, JPEG, PNG şi multe alte tipuri de imagini. Funcţiile folosite pentru crearea imaginilor pe baza

acestor tipuri de fişiere se numesc `ImageCreateFromGif()`, `ImageCreateFromJpg()`, `ImageCreateFromPng()` şi aşa mai departe. În exemplul următor puteţi vedea cât de uşor se creează o imagine nouă pe baza uneia deja existente. Imaginea de bază se găseşte în figura 13.5.

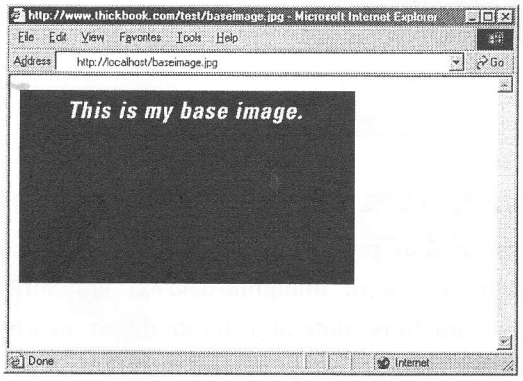


FIGURA 13.5. Imaginea de bază.

Listingul 13.5 vă arată cum să folosiţi o imagine deja existentă pe post de suprafaţă de desen, iar, apoi, desenează nişte elipse pe aceasta.

LISTINGUL 13.5. Crearea unei imagini noi pe baza uneia deja existente.

```
1:  <?
2:  //foloseste imaginea deja existenta drept suprafata de desenat
3:  $myImage = ImageCreateFromJpeg("baseimage.jpg");
4:
5:  //defineste culoarea alb
6:  $white = ImageColorAllocate($myImage, 255, 255, 255);
7:
8:  //deseneaza pe noua suprafata
9:  ImageFilledEllipse($myImage, 75, 70, 20, 20, $white);
10: ImageFilledEllipse($myImage, 150, 70, 20, 20, $white);
11: ImageFilledEllipse($myImage, 225, 70, 20, 20, $white);
12:
13: //afiseaza imaginea in browser
14: header ("Content-type: image/jpeg");
15: ImageJpeg($myImage);
16:
17: //curata suprafata de lucru
18: ImageDestroy($myImage);
19: ?>
```

Salvaţi listingul sub numele `imagefrombase.php` şi amplasaţi acest fişier în directorul rădăcină al serverului dumneavoastră Web. Atunci când este accesat, în browser trebuie să apară ceva similar cu figura 13.6.

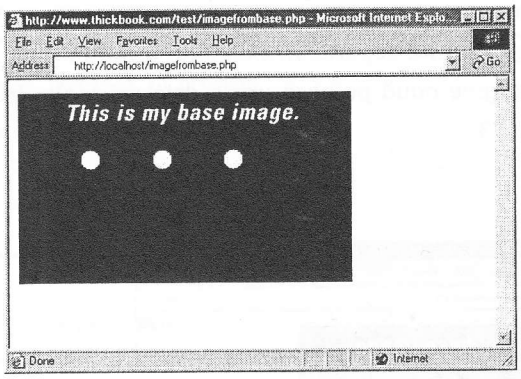


FIGURA 13.6. Desenarea pe o imagine deja existentă.

Următorul exemplu prezintă un proces şi mai avansat utilizând câteva funcţii de modificare a imaginilor. În acest caz, imaginile deja existente sunt patru fişiere PNG, conţinând câte un triunghi echilateral, colorate diferit, pe un fond gri. În listingul 13.6, punem aceste imagini una peste cealaltă şi le modificăm culorile în fiecare dintre paşi, astfel încât fundalul gri să devină transparent, iar imaginea de dedesubt să fie vizibilă.

LISTINGUL 13.6. Imaginile sunt plasate una peste cealaltă, fundalul fiecăreia devenind transparent.

```
1:  <?
2:  //foloseste imaginea deja existenta drept suprafata de desenat
3:  $baseimage = ImageCreateFromPng("img1.png");
4:
5:  //parcurge imaginile, de la a doua pana la ultima
6:  for($i=2; $i <5; $i++) {
7:    //defineste culoarea transparenta, si suprapune
8:    $myImage = ImageCreateFromPng("img".$i.".png");
9:    $gray = ImageColorAllocate($myImage, 185, 185, 185);
10:   ImageColorTransparent($myImage, $gray);
11:   ImageCopyMerge($baseimage,$myImage,0,0,0,0,150,150,100);
12: }
13:
14: //afiseaza imaginea in browser
15: header ("Content-type: image/png");
16: ImagePng($baseimage);
17:
18: //curata suprafata de lucru
19: ImageDestroy($baseimage);
20: ?>
```

Pe linia 3, una dintre imagini este selectată drept imagine de bază. În acest caz, este vorba despre `img1.png`. Cea mai mare parte a activităţii se desfăşoară în cadrul buclei for de pe liniile 8–11. Ştiind că avem patru imagini şi că prima dintre ele este deja folosită ca imagine de bază, mai rămân trei imagini care trebuie

aşezate peste aceasta şi făcute transparente. După ce se creează un strat nou de imagine pe linia 8, zonele sale gri sunt indicate ca fiind transparente şi este adăugat deasupra imaginii de bază. Pe măsură ce straturile sunt puse unul peste celălalt, imaginea de bază conţine din ce în ce mai multe straturi, până ce numărul total de straturi ajunge la patru. În acel moment, imaginea este trimisă către browser pe liniile 15–16.

Salvaţi listingul sub numele `imagestacker.php` şi amplasaţi acest fişier în directorul rădăcină al serverului dumneavoastră Web. Atunci când este accesat, în browser trebuie să apară ceva similar cu figura 13.7.

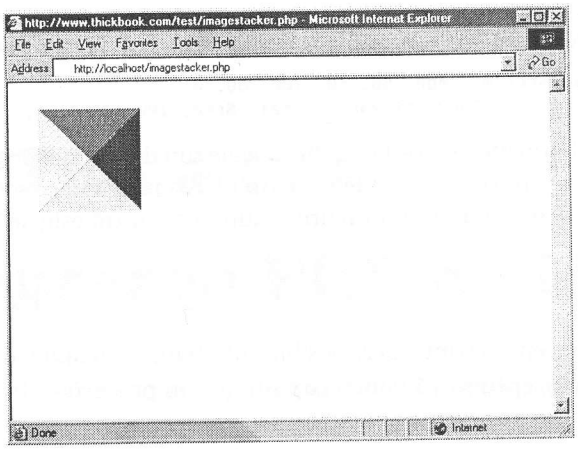


FIGURA 13.7. Imaginile transparente puse una peste cealaltă formează o imagine compusă.

Rezumat

Acest capitol v-a oferit o introducere scurtă despre ceea ce puteţi să faceţi cu imaginile în PHP. Vă recomand secţiunea intitulată *Image Functions* din Manualul PHP, aflată la adresa `http://www.php.net/image`, nu doar pentru că acolo există o listă completă a funcţiilor de manipulare a imaginilor, dar şi pentru că acolo puteţi găsi exemple şi discuţii despre modul în care acestea pot fi folosite în aplicaţiile dumneavoastră.

În cadrul acestei ore aţi învăţat să instalaţi şi utilizaţi biblioteci suplimentare pentru lucrul cu imaginile — şi anume pentru lucrul cu fişierele JPEG şi PNG. Aţi învăţat etapele de bază reprezentate de crearea unei suprafeţe de desen, de alocarea culorilor şi de desenarea şi umplerea formelor geometrice. Aţi învăţat că suprafaţa de desen poate să fie reprezentată de o imagine deja existentă, dar şi cum să suprapuneţi straturile de imagine pentru a obţine o imagine finală compusă, formată din straturile ce au fost suprapuse.

Întrebări şi răspunsuri

Î. Cum folosesc date dinamice pentru a crea feliile unui grafic de tip diagramă circulară?

R. Atunci când creaţi orice imagine, punctele de pornire şi lungimile liniilor pot fi indicate în mod dinamic — pot fi variabile ale căror valori să fie extrase dintr-o bază de date, introduse de utilizator, sau calculate în cadrul scriptului. De exemplu, această linie de cod creează un arc plin, de 90 de grade, de culoare roşie:

```
ImageFilledArc($myImage, 50, 50, 100, 50, 0, 90, $red, IMG_ARC_PIE);
```

Puteţi face în aşa fel încât arcul roşu din partea de sus a diagramei să reprezinte procentul vânzărilor din luna mai, care este conţinut de o variabilă denumită `$procent_vanzari_mai`. Linia de cod ar deveni ceva de genul:

```
ImageFilledArc($myImage, 50, 50, 100, 50, 0,
                $procent_vanzari_mai, $red, IMG_ARC_PIE);
```

Numărul provine, în acest caz, din calcule sau din interogări asupra unei baze de date pe care le efectuează scriptul. Asiguraţi-vă că scrieţi cod care să verifice că suma tuturor arcurilor dumneavoastră este 360.

Atelier

Atelierul este conceput să anticipeze posibile întrebări, să vă ajute să recapitulaţi cele învăţate şi să începeţi să vă puneţi cunoştinţele în practică.

Test

1. Ce valori RGB aţi folosi pentru a obţine negru pur şi alb pur?
2. Cum creaţi o suprafaţă de desen nouă, goală care să aibă 300 de pixeli în lăţime şi 200 de pixeli în înălţime?
3. Ce funcţie se foloseşte pentru a desena un poligon? Dar pentru a desena un poligon plin?

Soluţii

1. `(0,0,0)` şi `(255,255,255)`.
2. `$new_image = ImageCreate(300, 200);`
3. `ImagePolygon()` şi `ImageFilledPolygon()`.

Activitate

În loc să creaţi o diagramă circulară, folosiţi-vă talentul a desen pentru a crea un grafic cu bare, orizontale sau verticale, fiecare late de 30 de pixeli si haşurate cu aceeaşi culoare, dar de lungimi diferite. Asiguraţi-vă că între bare există un spaţiu de cel puţin 10 pixeli.

PARTEA a IV-a

Integrarea PHP cu MySQL

Capitolul 14. Procesul de proiectare a bazelor de date 261

Capitolul 15. Comenzi SQL de bază 273

Capitolul 16. Interacțiunea între scripturile MySQL şi PHP 321

CAPITOLUL 14
Procesul de proiectare a bazelor de date

În acest capitol, veţi învăţa procesul de gândire necesar în vederea proiectării unei baze de date relaţionale. După acest capitol axat pe teorie, vă veţi aventura direct să creaţi baze de date, învăţând comenzile MySQL esenţiale astfel încât să puteţi integra MySQL în aplicaţiile dumneavoastră proprii.

În acest capitol, veţi învăţa:

- unele avantaje ale proiectării corecte a bazelor de date;
- tipurile de relaţii între tabele;
- cum să normalizaţi o bază de date;
- cum să implementaţi un proces corespunzător de proiectare a bazelor de date.

Importanţa proiectării corecte a bazelor de date

Proiectarea corectă a bazelor de date este crucială pentru obţinerea unor aplicaţii de înaltă performanţă, în acelaşi fel în care profilul aerodinamic este important pentru o maşină de curse. Dacă maşina nu are laturi rotunjite, ea va produce prea multă frecare cu aerul şi va merge mai încet. Acelaşi lucru este valabil şi în cazul bazelor de date. Dacă o bază de date nu conţine relaţii optimizate — dacă nu este, adică, normalizată — ea nu va putea fi utilizată cu un maxim de eficienţă.

În afară de problema performanţei, mai există şi problema întreţinerii. Baza dumneavoastră de date ar trebui să fie uşor de întreţinut. Pentru aceasta, este imperativ ca baza de date să stocheze cât mai puţine informaţii redundante (sau deloc). Dacă aveţi multe date redundante şi un element dintre aceste date se modifică (cum ar fi, de exemplu, cazul schimbării unui nume), respectiva modificare ar trebui aplicată în toate locurile în care apare acel element. Pentru a elimina înregistrările duble, va trebui să creaţi o tabelă a valorilor posibile şi să folosiţi o cheie cu ajutorul căreia să creaţi referinţe către acele valori. În acest fel, dacă valorile se schimbă, modificarea va fi efectuată o singură dată — în tabela principală.

De exemplu, să presupunem că sunteţi responsabil de întreţinerea unei baze de date cu studenţi şi cursurile la care ei sunt înscrişi. Dacă 35 dintre aceşti studenţi sunt înscrişi la un acelaşi curs, denumit Matematici Avansate, numele acestui curs ar apărea de 35 de ori în tabelă. Acum, să presupunem că profesorul se hotărăşte să schimbe numele cursului, spunându-i Matematică IV. În acest caz, va trebui să schimbaţi 35 de înregistrări pentru ca ele să reflecte schimbarea numelui cursului. Dacă baza de date ar fi proiectată astfel încât numele cursurilor să apară într-o tabelă, iar în înregistrările studenţilor să fie stocate doar numerele de identificare ale cursurilor, nu ar mai trebui să schimbaţi decât o singură înregistrare, nu 35, pentru a actualiza baza de date în urma schimbării numelui cursului.

Beneficiile unei baze de date planificate şi proiectate corect sunt numeroase şi, evident, cu cât munciţi mai mult la început, cu atât mai puţin va trebui să munciţi mai apoi. După ce aplicaţia care o utilizează a fost lansată, este deja târziu să modificaţi proiectul bazei de date şi, în caz că acest lucru se întâmplă — şi pot să vă asigur că se întâmplă — costurile implicate sunt foarte mari.

Din aceste motive, trebuie să petreceţi mult timp proiectând baza dumneavoastră de date înainte să începeţi să scrieţi o aplicaţie. În restul acestui capitol, veţi învăţa mai multe despre relaţii şi normalizare, două dintre cele mai importante piese ale puzzle-ului reprezentat de proiectarea bazelor de date.

Tipuri de relaţii între tabele

Relaţiile dintre tabele pot să fie de mai multe feluri:

- relaţii unul la unul;
- relaţii unul la mai multe;
- relaţii mai multe la mai multe.

De exemplu, să presupunem că aveţi o tabelă denumită `angajati` care conţine codul numeric personal, numele şi departamentul în care lucrează fiecare angajat. Să presupunem că mai aveţi o tabelă separată, `departamente`, cu toate departamentele existente, conţinând, pentru fiecare dintre ele, un ID şi numele departamentului. În tabela angajati, câmpul ID `Departament` se potriveşte cu un ID din tabela departamente. Puteţi vedea acest tip de relaţie în figura 14.1. Simbolul PK din dreptul numelui unui câmp indică faptul că respectivul câmp este o cheie primară (primary key).

În secţiunile următoare, vom analiza mai îndeaproape fiecare dintre tipurile de relaţii.

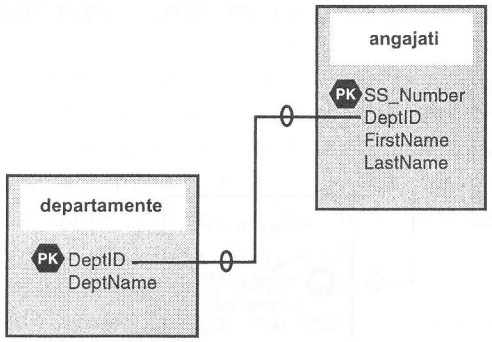


FIGURA 14.1. Tabelele `angajati` şi `departamente` sunt conectate prin intermediul cheii `DeptID`.

Relaţii unul la unul

În cadrul unei relaţii de tip unul la unul, o cheie dintr-o tabelă apare o singură dată într-o tabelă cu care aceasta se află în relaţie. Tabelele `angajati` şi `departamente` nu au o relaţie de tip unul la unul deoarece, fără îndoială, mai mulţi angajaţi aparţin aceluiaşi departament. O relaţie unul la unul ar exista, de exemplu, dacă fiecărui angajat i s-ar aloca un singur calculator în cadrul companiei. Figura 14.2 arată o relaţie unul la unul între angajaţi şi calculatoare.

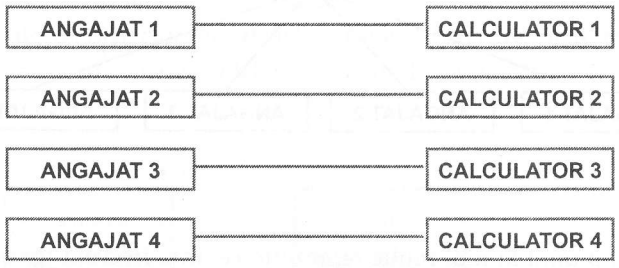


FIGURA 14.2. Fiecărui angajat îi este alocat câte un calculator.

Tabelele `angajati` şi `calculatoare` din baza dumneavoastră de date arată ca în figura 14.3, care ilustrează o relaţie unul la unul.

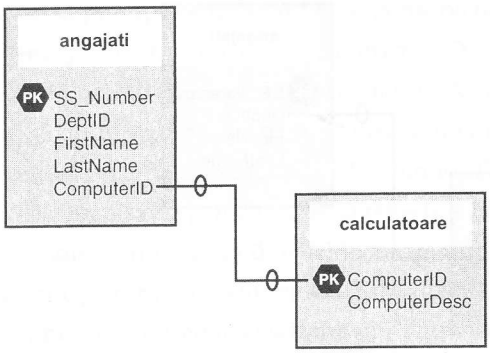


FIGURA 14.3. Relația unul la unul în modelul de date.

Relații de tip unul la mai multe

Într-o relație de tip unul la mai multe, cheile dintr-o tabelă apar de mai multe ori într-o tabelă cu care acesta se găsește în relație. Exemplul din figura 14.1, care reprezintă relația dintre angajați și departamente, ilustrează o relație de tip unul la mai multe. Un exemplu mai apropiat de viața reală ar fi diagrama organizațională a departamentului, așa cum se vede în figura 14.4.

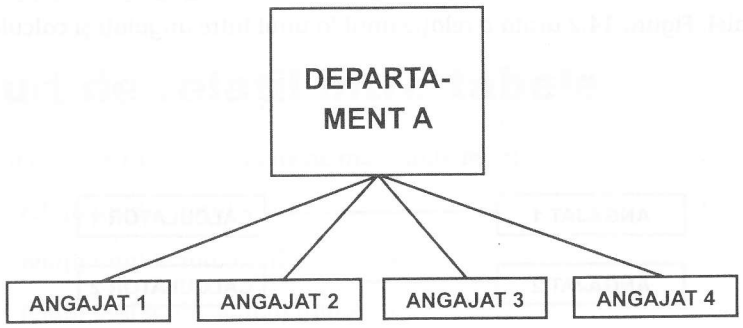


FIGURA 14.4. Un departament conține mai mulți angajați.

Relația de tip unul la mai multe reprezintă cel mai frecvent tip de relație. Un alt exemplu practic ar fi reprezentat de folosirea abrevierii numelor de județe într-o bază de date cu adrese; fiecare județ are un identificator unic (PH pentru Prahova, BZ pentru Buzău și așa mai departe), iar fiecare adresă din România are un județ asociat.

Dacă aveți opt prieteni în Prahova și cinci în Buzău, va trebui să folosiți numai două denumiri abreviate de județe în tabela dumneavoastră. Una dintre abrevieri reprezintă o relație unu la opt (PH), iar cealaltă reprezintă o relație unu la cinci (BZ).

Relații de tipul mai multe la mai multe

Relaţiile de tipul mai multe la mai multe provoacă probleme în exemplele practice de baze de date normalizate într-o aşa de mare măsură încât este uzual ca aceste relaţii să fie sparte în serii de relaţii unul la mai multe. În cadrul unei relaţii de tipul mai multe la mai multe, valoarea cheie dintr-o tabelă poate apărea de mai multe ori într-o tabelă cu care aceasta se află în relaţie. Până acum, seamănă cu o relaţie de tipul unul la mai multe, dar lucrurile nu se opresc aici: şi reciproca este adevărată, adică o cheie primară din cea de-a doua tabelă poate, de asemenea, să apară de mai multe ori în prima tabelă.

Gândiţi-vă la o astfel de relaţie folosind exemplul cu studenţii şi cursurile. Un student are un ID şi un nume. Un curs are un ID şi un nume. Un student, de obicei, este înscris la mai multe cursuri în acelaşi timp, iar un curs conţine, întotdeauna mai mult decât un student, aşa cum puteţi vedea în figura 14.5.

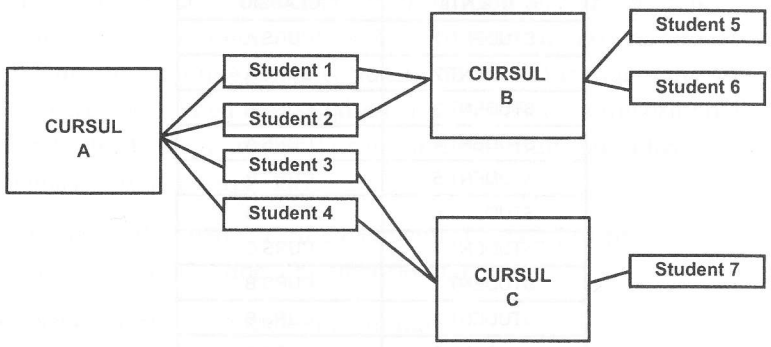


FIGURA 14.5. Studenții sunt înscriși la mai multe cursuri, iar cursurile conțin, fiecare, mai mulți studenți.

După cum puteţi vedea, în cazul acestui tip de relaţie nu avem la dispoziţie o metodă facilă de conectare a tabelelor. Tabelele noastre, ar putea arăta ca în figura 14.6, neavând, aparent, nici o relaţie unul cu celălalt.

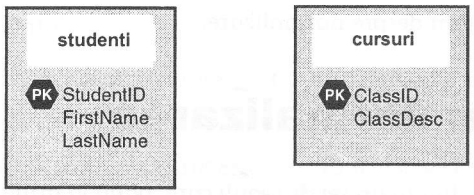


FIGURA 14.6. Tabela `studenti` şi tabela `cursuri`, fără relație între ele.

Pentru a crea relaţia teoretică de tip mai multe la mai multe, va trebui să creaţi o tabelă intermediară, una care să se găsească între cele două tabele şi, practic, să le cupleze. Aţi putea construi ceva similar cu tabela din figura 14.7.

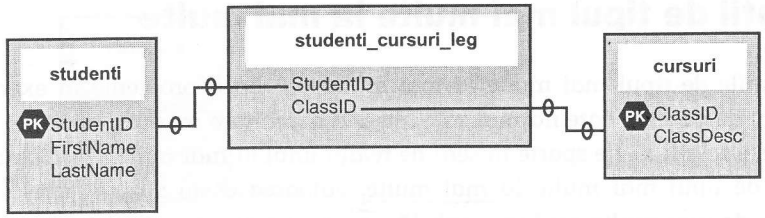


FIGURA 14.7. Tabela `studenti_cursuri_leg` joacă rolul de intermediar.

Dacă luaţi informaţia din figura 14.5 şi o puneţi în tabelul intermediar, veţi avea ceva similar cu figura 14.8.

FIGURA 14.8. Tabela `studenti_cursuri_leg`, populată cu date.

STUDENTID	CLASSID
STUDENT 1	CURS A
STUDENT 2	CURS A
STUDENT 3	CURS A
STUDENT 4	CURS A
STUDENT 5	CURS B
STUDENT 6	CURS B
STUDENT 7	CURS C
STUDENT 1	CURS B
STUDENT 2	CURS B
STUDENT 3	CURS C
STUDENT 4	CURS C

După cum puteţi vedea, mai mulţi studenţi şi mai multe cursuri coexistă fericite în cadrul tabelei `studenti_cursuri_leg`.

După această introducere asupra tipurilor de relaţii între tabele, ar trebui să ne fie floare la ureche să învăţăm despre normalizare.

Să înţelegem normalizarea

Normalizarea este pur şi simplu un set de reguli care, până la urmă, vă vor face viaţa mai uşoară atunci când veţi juca rolul de administrator de baze de date. Normalizarea reprezintă arta organizării bazelor de date în aşa fel încât tabelele să fie relaţionate în felul cel mai potrivit, formând o structură flexibilă în vederea extinderilor ulterioare.

Seturile de reguli folosite în normalizare se numesc forme normale. Dacă proiectul unei baze de date respectă primul set de reguli, se spune despre respectiva bază de date că este în prima formă normală. Dacă primele trei seturi de reguli de normalizare sunt respectate, se spune că baza de date este în a treia formă normală.

De-a lungul acestui capitol, veţi învăţa despre primele trei reguli de normalizare şi, să sperăm, le veţi respecta atunci când vă veţi crea propriile aplicaţii. Veţi folosi un set de tabele exemplificatoare pentru o bază de date cu studenţi şi cursuri, pe care o veţi aduce în a treia formă normală.

Problemele tabelei bidimensionale

Înainte să ne lansăm în explicaţii despre prima formă normală, trebuie să începem de la ceva ce trebuie reparat. În cazul unei baze de date, este vorba despre tabela bidimensională. O tabelă bidimensională este ca o foaie de calcul tabelar — are multe, multe coloane. Nu există relaţii între mai multe tabele; toate datele pe care le-aţi putea dori se găsesc aici, în tabela bidimensională. Această situaţie nu este eficientă, iar tabela consumă mai mult spaţiu fizic pe discul dur decât o bază de date normalizată.

În cazul bazei dumneavoastră cu studenţi şi cursuri, să presupunem că aveţi următoarele câmpuri în tabela bidimensională:

- `NumeStudent` — numele studentului;
- `IDCurs1` — ID-ul primului curs la care este înscris studentul;
- `DescriereCurs1` — Descrierea primului curs la care este înscris studentul;
- `ProfesorCurs1` — Profesorul de la primul curs la care este înscris studentul;
- `IDCurs2` — ID-ul celui de-al doilea curs la care este înscris studentul;
- `DescriereCurs2` — Descrierea celui de-al doilea curs la care este înscris studentul;
- `ProfesorCurs2` — Profesorul de la cel de-al doilea curs la care este înscris studentul;
- Repetaţi coloanele `IDCurs`, `DescriereCurs` şi `ProfesorCurs` de multe ori pentru a conţine toate cursurile pe care le pot urma studenţii în cursul carierei lor academice.

Pe baza a ce aţi învăţat până acum, ar trebui să puteţi identifica prima problemă: coloanele `IDCurs`, `DescriereCurs` şi `ProfesorCurs` sunt grupuri care se repetă.

Eliminarea redundanţei este primul pas către normalizare, aşa că, în cele ce urmează, vom trece această tabelă în prima formă normală. Dacă tabela ar rămâne în forma în care se află, ar fi foarte mult spaţiu ocupat fără a fi folosit şi, pe de altă parte, foarte mult spaţiu ar fi folosit degeaba — avem de-a face cu o tabelă cât se poate de ineficientă.

Prima formă normală

Regulile pentru prima formă normală sunt după cum urmează:

- eliminarea informaţiei redundante;
- crearea de tabele separate pentru datele aflate în relaţie.

Dacă vă gândiţi la structura tabelei de mai sus pentru baza de date cu studenţi şi cursuri, cu multe seturi de câmpuri repetate, atunci puteţi să identificaţi două subiecte distincte: studenţi şi cursuri. Pentru a aduce această bază de date în prima formă normală, trebuie să creaţi două tabele: unul pentru studenţi şi altul pentru cursuri, aşa cum se poate vedea în figura 14.9.

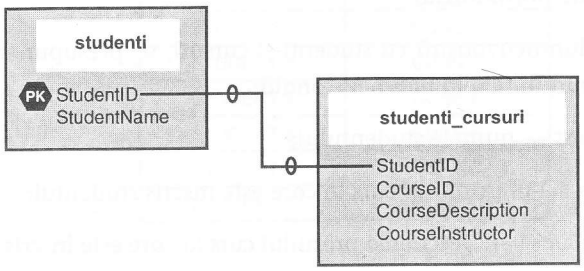


FIGURA 14.9. Spargerea tabelei plane în două tabele.

Cele două tabele reprezintă acum o relaţie de tip unul la mai multe, un student fiind înscris la mai multe cursuri. Studenţii pot să se înscrie la oricâte cursuri vor şi nu sunt limitaţi de numărul grupurilor de coloane `IDCurs/DescriereCurs/ProfesorCurs` care există în tabela plană.

Următorul pas este aducerea tabelelor în cea de-a doua formă normală.

Cea de-a doua formă normală

Regula pentru cea de-a doua formă normală este:

- nici un atribut non-cheie nu depinde de o porţiune a cheii primare.

Mai pe înţeles, acest lucru înseamnă că, în cazul în care există câmpuri care nu au o legătură foarte mare cu cheia primară în tabela dumneavoastră, mai aveţi de lucru. În exemplul cu studenţii şi cursurile, trebuie să puneţi cursurile separat într-o tabelă proprie şi să modificaţi tabela `studenti_cursuri`.

`IDCurs`, `DescriereCurs` şi `ProfesrCurs` pot deveni câmpuri într-o tabelă de sine stătătoare, având cheia primară `IDCurs`. Tabela studenti_cursuri trebuie, atunci, să conţină două câmpuri: `IDStudent` şi `IDCurs`. Puteţi vedea noua formă a proiectului în figura 14.10.

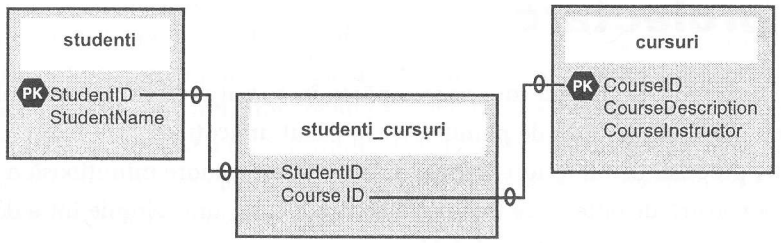


FIGURA 14.10. Tabelele dumneavoastră aduse în cea de-a doua formă normală.

Această structură ar trebui să vă fie cunoscută, fiind vorba despre o relaţie de tip mai multe la mai multe, implementată cu ajutorul unei tabele de legătură. Cea de-a treia formă normală este ultima pe care o vom discuta aici şi veţi descoperi că ea este la fel de uşor de înţeles ca primele două.

Cea de-a treia formă normală

Regula pentru cea de-a treia formă normală este:

- nici un atribut nu depinde de un alt atribut care nu este cheie primară.

Acest lucru înseamnă, de fapt, că trebuie să vă uitaţi cu atenţie la tabelele dumneavoastră şi să vedeţi dacă există câmpuri care mai pot fi puse în tabele separate şi care nu depind de cheia primară. Gândiţi-vă ce date se repetă şi veţi descoperi răspunsul — profesorii. Inevitabil, un profesor va preda mai multe cursuri. Cu toate acestea, `f` nu este cheie. Astfel, dacă scoateţi acest câmp şi creaţi o tabelă nouă doar în scopul optimizării eficienţei şi uşurinţei de întreţinere (aşa cum este arătat în figura 14.11), veţi obţine cea de-a treia formă normală.

Cea de-a treia formă normală este cea mai potrivită, de obicei, pentru eliminarea redundanţelor şi permiterea flexibilităţii şi scalabilităţii. Următoarea secţiune vă va da indicaţii despre cum trebuie să gândiţi când proiectaţi o bază de date şi cum se integrează procesul acesta în cadrul proiectării unei aplicaţii.

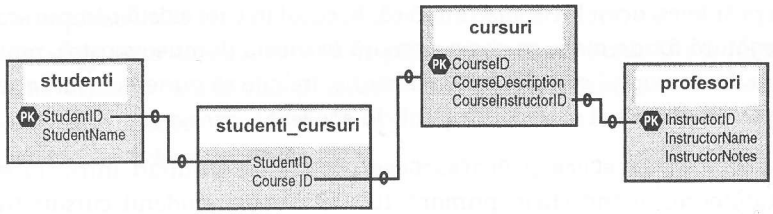


FIGURA 14.11. Aceleaşi tabele, aduse în cea de-a treia formă normală.

Implementarea procesului de proiectare

Cea mai mare problemă care poate apărea în cursul proiectării unei aplicaţii este reprezentată de lipsa de planificare. În cazul aplicaţiilor care folosesc baze de date, procesul de proiectare trebuie să includă o evaluare minuţioasă a bazei dumneavoastră de date — ce trebuie să conţină, care sunt relaţiile între date şi, cel mai important, dacă este scalabilă.

În general, etapele procesului de proiectare sunt:

- definirea obiectivului;
- proiectarea structurilor de date (tabele, câmpuri);
- evidenţierea relaţiilor;
- definirea şi implementarea regulilor de utilizare;
- crearea aplicaţiei.

Crearea aplicaţiei este ultima etapă — în nici un caz prima! Mulţi dezvoltatori au o idee de aplicaţie, o construiesc şi, abia apoi, se întorc şi încearcă să îi potrivească un set de tabele cu date. Această abordare este complet de-a-ndoaselea, ineficientă şi va costa foarte mult atât din punct de vedere al timpului cât şi al banilor.

Înainte de a începe procesul de proiectare al oricărei aplicaţii, încercaţi să o povestiţi. Dacă nu puteţi să descrieţi aplicaţia — inclusiv obiectivele, audienţa şi segmentul de piaţă — atunci nu sunteţi pregătit să o construiţi şi, cu atât mai puţin, să modelaţi baza de date.

Abia după ce puteţi să descrieţi acţiunile şi nuanţele aplicaţiei dumneavoastră altor oameni astfel încât ei să le înţeleagă, puteţi să începeţi să vă gândiţi la tabelele pe care trebuie să le creaţi. Începeţi cu o tabelă bidimensională mare pentru că, după ce o scrieţi, puteţi să vă folosiţi abilităţile de normalizare proaspăt dobândite. Veţi putea să descoperiţi redundanţele şi să vizualizaţi relaţiile.

Următorul pas este normalizarea propriu-zisă. Treceţi tabela bidimensională în prima formă normală şi continuaţi aşa până la cea de-a treia, dacă este posibil. Folosiţi hârtie şi creion, bileţele lipite de monitor sau orice v-ar putea ajuta să vizualizaţi tabelele şi relaţiile. Nu este nici o ruşine să modelezi baza de date pe hârtie până când eşti gata să creezi tabelele de-adevărat. În plus, hârtia şi mina de pix sunt mai ieftine decât instrumentele software care ar putea să facă aceste lucruri în locul dumneavoastră; programele de modelare costă între câteva sute şi câteva mii de dolari!

În momentul în care aveţi un model de date preliminar, priviţi-l din punctul de vedere al aplicaţiei. Mai bine, chiar, priviţi-l din punctul de vedere al persoanei care ar utiliza aplicaţia pe care o construiţi. Acesta este momentul în care definiţi regulile de utilizare şi vedeţi dacă modelul de date la care aţi ajuns ar rezista în condiţii reale. Un exemplu de regulă de utilizare pentru o aplicaţie de înregistrare online ar fi: „Fiecare utilizator trebuie să aibă o adresă de e-mail şi aceasta nu trebuie să îi aparţină şi altui utilizator". Dacă datele din câmpul `AdresaEmail` nu ar fi unice în cadrul modelului dumneavoastră de date, ar însemna că modelul nu rezistă la aplicarea acestei reguli.

Numai şi numai după ce regulile de utilizare au fost aplicate modelului de date, puteţi începe să programaţi aplicaţia. Puteţi fi sigur că modelul de date este solid şi că, programând, nu vă veţi trezi pe un drum înfundat. Acest lucru se întâmplă mult prea des în viaţa de zi cu zi.

Rezumat

Implementarea unui proces adecvat de proiectare a bazelor de date este singura modalitate prin care puteţi scrie aplicaţii eficiente, flexibile şi uşor de gestionat şi întreţinut. Un aspect important al proiectării bazelor de date este folosirea relaţiilor între tabele în locul utilizării unui fişier bidimensional mare care să conţină toate datele. Relaţiile pot fi de mai multe tipuri, printre care unul la unul, unul la mai multe şi mai multe la mai multe.

Folosirea relaţiilor pentru organizarea corespunzătoare a datelor dumneavoastră se numeşte normalizare. Există multe niveluri de normalizare, dar cele principale sunt prima, a doua şi a treia formă normală. Fiecare nivel are o regulă sau două care trebuie respectate. Respectarea tuturor regulilor vă ajută să vă asiguraţi că baza de date este bine organizată şi flexibilă.

Pentru a duce o idee din faza iniţială până la cea finală, trebuie să urmaţi un proces de proiectare. În esenţă, acest proces înseamnă „gândiţi înainte să acţionaţi". Discutaţi regulile, cerinţele şi obiectivele; apoi creaţi o versiune finală a tabelelor dumneavoastră normalizate.

Întrebări şi răspunsuri

Î. Există numai trei forme normale?

R. Nu, există mai multe forme normale. Celelalte forme sunt forma normală Boyce-Codd, cea de-a patra formă normală şi cea de-a cincea formă normală/forma normală joncţiune–proiecţie. Aceste forme nu sunt respectate, în general, pentru că beneficiile de eficienţă pe care le aduc nu compensează costul şi munca de care este nevoie pentru a le respecta.

Atelier

Atelierul este conceput să anticipeze posibile întrebări, să vă ajute să recapitulaţi cele învăţate şi să începeţi să vă puneţi cunoştinţele în practică.

Test

1. Numiţi trei tipuri de relaţii între date.
2. Relaţiile de tip mai multe la mai multe sunt dificil de implementat într-un proiect eficient de bază de date. Ce este de făcut, în acest caz?

Soluţii

1. Unul la unul, unul la mai multe şi mai multe la mai multe.
2. Creăm o serie de relaţii de tip unul la mai multe folosind tabele intermediare de legătură.

Activitate

Explicaţi fiecare dintre cele trei forme normale unei persoane care lucrează cu tabele bidimensionale şi foi de calcul tabelar.

CAPITOLUL 15
Comenzi SQL de bază

Din moment ce tocmai am învăţat fundamentele procesului de proiectare a bazelor de date, acest capitol prezintă pe scurt elementele sintactice de bază din SQL, pe care le veţi folosi pentru a crea şi manipula bazele de date în MySQL. Acest capitol este foarte orientat pe activitatea practică şi porneşte de la premisa că aveţi posibilitatea de a trimite interogări prin intermediul monitorului MySQL, fie sub Windows sau sub Linux/Unix. Dacă folosiţi o interfaţă grafică pentru MySQL, se presupune că ştiţi cum să trimiteţi interogări cu ajutorul respectivei interfeţe.

În acest capitol, veţi învăţa:

- cele mai importante tipuri de date din MySQL;
- cum să folosiţi comanda `CREATE TABLE` pentru a crea o tabelă;
- cum să folosiţi comanda `INSERT` pentru a introduce înregistrări;
- cum să folosiţi comanda `SELECT` pentru a regăsi înregistrări;
- cum să folosiţi funcţiile de bază, clauza `WHERE` şi clauza `GROUP BY` în expresiile `SELECT`;
- cum să selectaţi din mai multe tabele, folosind `JOIN`;
- cum să folosiţi comenzile `UPDATE` şi `REPLACE` pentru a modifica înregistrări existente;
- cum să folosiţi comanda `DELETE` pentru a elimina înregistrări;
- cum să folosiţi funcţiile de manipulare a şirurilor de caractere ce sunt predefinite în MySQL;
- cum să folosiţi funcţiile predefinite în MySQL ce operează cu date calendaristice şi ore.

Tipurile de date în MySQL

Definirea corespunzătoare a câmpurilor dintr-o tabelă este importantă pentru gradul final de optimizare a bazei dumneavoastră de date. Ar trebui să folosiţi numai tipurile şi dimensiunile de câmp de care aveţi cu adevărat nevoie; nu definiţi un câmp de 10 caractere dacă ştiţi că veţi folosi numai 2 caractere. Aceste tipuri de câmpuri (sau coloane) se mai numesc şi tipuri de date, în sensul că se referă la datele pe care urmează să le stocaţi în aceste câmpuri.

MySQL foloseşte multe tipuri de date diferite, iar acestea se împart în trei categorii: numerice, calendaristice şi şiruri de caractere. Fiţi foarte atenţi pentru că definirea tipurilor de date este mai importantă decât orice altă etapă a procesului de creare a tabelelor.

Tipuri de date numerice

MySQL foloseşte toate tipurile de date numerice standard din ANSI SQL, aşa că dacă aţi utilizat înainte un alt sistem de baze de date, aceste definiţii vi se vor părea cunoscute. Următoarea listă enumeră tipurile de date numerice frecvent folosite şi descrierile acestora:

Apropo

> În această listă a tipurilor de date numerice vor fi folosite exprimările cu semn şi fără semn. Dacă vă aduceţi aminte algebra elementară, vă veţi da seama că un număr întreg cu semn este un număr întreg pozitiv sau negativ, în timp ce un număr întreg fără semn este un număr întreg non-negativ.

- `INT` — Un număr întreg de dimensiuni normale care poate fi cu semn sau fără semn. Dacă este cu semn, atunci intervalul permis este între –2147483648 şi 2147483647. Dacă este fără semn, atunci intervalul permis este între 0 şi 4294967295. Puteţi specifica o lungime de până la 11 cifre.
- `TINYINT` — Un număr întreg foarte mic ce poate fi cu semn sau fără semn. Dacă este cu semn, intervalul permis este între –128 şi 127. Dacă este fără semn, intervalul permis este între 0 şi 255. Puteţi specifica o lungime de până la 4 cifre.
- `SMALLINT` — Un număr întreg mic ce poate fi cu semn sau fără semn. Dacă este cu semn, intervalul permis este între –32768 şi 32767. Dacă este fără semn, intervalul permis este între 0 şi 65535. Puteţi specifica o lungime de până la 5 cifre.
- `MEDIUMINT` — Un număr întreg de dimensiuni medii care poate fi cu semn sau fără semn. Dacă este cu semn, intervalul permis este între –8388608 şi 8388607. Dacă este fără semn, intervalul permis este între 0 şi 16777215. Puteţi specifica o lungime de până la 9 cifre.
- `BIGINT` — Un număr întreg mare ce poate fi cu semn sau fără semn. Dacă numărul este cu semn, intervalul permis este între –9223372036854775808 şi 9223372036854775807. Dacă este fără semn, intervalul permis este între 0 şi 18446744073709551615. Puteţi specifica o lungime de până la 11 cifre.
- `FLOAT(M,D)` — Un număr real în virgulă mobilă care nu poate fi fără semn. Puteţi defini dimensiunea de afişare (`M`) şi numărul de zecimale (`D`). Acestea nu sunt obligatorii şi au valorile implicite `10,2`, unde `2` este numărul de zecimale, iar `10` este numărul total de cifre (inclusiv zecimalele). Precizia poate ajunge până la 24 de zecimale pentru `FLOAT`.

- `DOUBLE(M,D)` — Un număr real în virgulă mobilă cu precizie dublă şi care nu poate fi fără semn. Puteţi defini lungimea de afişare (`M`) şi numărul de zecimale (`D`). Acestea nu sunt obligatorii şi au valorile implicite `16,4`, unde `4` este numărul de zecimale. Precizia poate ajunge până la `53` de zecimale pentru un număr `DOUBLE`. `REAL` este un sinonim al tipului de date `DOUBLE`.
- `DECIMAL(M,D)` — Un număr real în virgulă mobilă despachetat care nu poate fi fără semn. Într-un număr despachetat, fiecare zecimală corespunde unui octet. Definirea lungimii de afişare (`M`) şi a numărului de zecimale (`D`) este obligatorie. `NUMERIC` este un sinonim al tipului de date `DECIMAL`.

Dintre toate tipurile de date MySQL, cel mai des veţi folosi, probabil, tipul `INT`. Puteţi avea probleme dacă definiţi câmpuri mai mici decât este necesar; de exemplu, dacă definiţi un câmp `ID` ca fiind de tipul `TINYINT` fără semn, nu veţi putea să inseraţi cu succes o a 255-a înregistrare dacă `ID`-ul este cheie primară (şi, de aceea, obligatoriu).

Tipuri calendaristice de date

MySQL are câteva tipuri de date disponibile pentru stocarea datelor calendaristice şi a orelor, iar aceste tipuri de date sunt flexibile din punct de vedere al introducerii de date. Cu alte cuvinte, puteţi introduce date care nu sunt cu adevărat zile, cum ar fi 30 februarie — februarie având numai 28 sau 29 de zile, dar niciodată 30. De asemenea, puteţi stoca date calendaristice cu informaţii lipsă. Dacă ştiţi că cineva s-a născut cândva în noiembrie 1980, puteţi folosi valoarea 1980-11-00, unde „00" ar fi fost ziua, dacă aţi fi ştiut-o.

Flexibilitatea tipurilor calendaristice din MySQL înseamnă şi că responsabilitatea de verificare a datelor cade în sarcina programatorului aplicaţiei. MySQL verifică doar două elemente de validitate: ca luna să fie între 0 şi 12 şi ca ziua să fie între 0 şi 31. MySQL nu verifică automat dacă a 30-a zi a celei de-a doua luni (30 februarie) este o dată validă.

Tipurile calendaristice de date din MySQL sunt:

- `DATE` — o dată calendaristică în format `AAAA-LL-ZZ`, cuprinsă între `1000-01-01` şi `9999-12-31`. De exemplu, data de 30 decembrie 1973 va fi stocată sub forma valorii `1973-12-30`.
- `DATETIME` — O combinaţie de dată şi oră în formatul `AAAA-LL-ZZHH:MM:SS`, cuprinsă între `1000-01-0100:00:00` şi `9999-12-3123:59:59`. De exemplu, ora 3:30 după amiază din ziua de 30 decembrie 1973 va fi stocată sub forma valorii `1973-12-3015:30:00`.
- `TIMESTAMP` — Un *timestamp* (marcă temporală) cuprins între miezul nopţii din data de 1 ianuarie 1970 şi cândva în anul 2037. Puteţi defini mai multe lungimi pentru un câmp de tip `TIMESTAMP`, care se corelează direct cu ce anume se stochează în respectivul câmp. Lungimea implicită

pentru TIMESTAMP este 14, care stochează AAALLZZHHMMSS. Acest format arată la fel ca formatul anterior, DATETIME, numai că nu există cratimele dintre numere; ora 3:30 din după amiaza zilei de 30 decembrie 1973 va fi stocată sub forma 19731230153000. Alte definiţii pentru TIMESTAMP sunt 12 (AALLZZHHMMSS), 8 (AAAALLZZ) şi 6 (AALLZZ).

- TIME — Stochează ora în formatul HH:MM:SS.
- YEAR(M) — Stochează anul în format de 2 sau 4 cifre. Dacă lungimea specificată este 2 (de exemplu, YEAR(2)), YEAR poate fi între 1970 şi 2069 (valori de la 70 la 69). Dacă lungimea specificată este 4, YEAR poate fi între 1901 şi 2155. Lungimea implicită este 4.

De cele mai multe ori, probabil, veţi folosi tipurile DATETIME şi DATE, iar pe celelalte le veţi folosi destul de rar.

Tipuri de şiruri de caractere

Deşi tipurile numerice şi cele calendaristice sunt interesante, cele mai multe date vor fi stocate în format de şir de caractere. Această listă descrie cele mai frecvent utilizate tipuri de şiruri de caractere din MySQL.

- CHAR(M) — Un şir de caractere cu lungime fixă cuprinsă între 1 şi 255 de caractere (de exemplu, CHAR(5)), completat la dreapta cu spaţii până la lungimea specificată spre a fi stocată. Definirea unei lungimi nu este obligatorie, dar valoarea implicită este 1.
- VARCHAR(M) — Un şir de caractere cu lungime variabilă cuprinsă între 1 şi 255 de caractere; de exemplu, VARCHAR(2). Trebuie să definiţi o lungime atunci când creaţi un câmp de tip VARCHAR.
- BLOB sau TEXT — Un câmp cu lungimea maximă de 65535 de caractere. BLOB-urile sunt „Binary Large Objects" (obiecte binare mari) şi sunt folosite pentru stocarea unor cantităţi mari de date în format binar, cum ar fi imaginile şi alte tipuri de fişiere. Câmpurile definite drept TEXT conţin, de asemenea, cantităţi mari de date; diferenţa dintre cele două este că sortările şi comparaţiile sunt dependente de capitalizare în cazul BLOB-urilor şi nu depind de capitalizare în cazul câmpurilor de tip TEXT. Nu trebuie să specificaţi o lungime când definiţi câmpuri de tip BLOB sau TEXT.
- TINYBLOB sau TINYTEXT — Un câmp de tip BLOB sau TEXT cu o lungime maximă de 255 de caractere. Nu trebuie să specificaţi o lungime când definiţi câmpuri de tip TINYBLOB sau TINYTEXT.
- MEDIUMBLOB sau MEDIUMTEXT — Un câmp de tip BLOB sau TEXT cu o lungime maximă de 16777215 caractere. Nu trebuie să specificaţi o lungime când definiţi câmpuri de tip MEDIUMBLOB sau MEDIUMTEXT.
- LONGBLOB sau LONGTEXT — Un câmp de tip BLOB sau TEXT cu o lungime maximă de 4294967295 de caractere. Nu trebuie să specificaţi o lungime când definiţi câmpuri de tip LONGBLOB sau LONGTEXT.

- ENUM — O enumerare, adică o listă. Când definiţi un câmp de tip ENUM, creaţi o listă de elemente dintre care trebuie selectată valoarea. De exemplu, dacă doriţi ca în câmpul dumneavoastră să fie stocat "A" sau "B" sau "C", veţi defini câmpul dumneavoastră sub forma ENUM('A','B','C') şi numai aceste valori (sau valoarea NULL) vor putea popula vreodată câmpul respectiv. ENUM-urile folosesc un index în vederea stocării elementelor.

Apropo

Tipul SET este similar cu ENUM prin aceea că se defineşte ca o listă. Diferenţa constă în faptul că în câmpurile de tip SET este stocată valoarea cu totul, nu doar indexul valorii respective, ca în cazul tipului ENUM.

Probabil că veţi folosi câmpuri de tip VARCHAR şi TEXT mult mai frecvent decât veţi folosi restul tipurilor, dar şi câmpurile de tip ENUM se vor dovedi utile.

Sintaxa de creare a unei tabele

Comanda de creare a unei tabele are nevoie de:

- numele tabelei;
- numele câmpurilor;
- definiţiile pentru fiecare câmp.

Sintaxa generică pentru creare unei tabele este:

```
CREATE TABLE nume_tabela(nume_coloana tip_coloana);
```

Bineînţeles că puteţi să daţi orice nume doriţi unei tabele, dar ar trebui să daţi nume care să reflecte utilitatea tabelelor respective. De exemplu, dacă aveţi o tabelă care stochează inventarul unei băcănii, probabil că nu este o idee bună să îi daţi tabelei numele s. Probabil că ar trebui să folosiţi ceva de genul grocery_inventory (inventar_bacanie). Tot aşa, numele câmpurilor ar trebui să fie cât de concise posibil, dar relevante pentru funcţia pe care o serveşte fiecare şi pentru datele stocate. De exemplu, aţi putea să denumiţi un câmp care stochează numele unui element item_name (nume_element), nu n.

Exemplul următor creează o tabelă generică grocery_inventory cu câmpuri pentru ID, nume, descriere, preţ şi cantitate:

```
mysql> CREATE TABLE grocery_inventory (
    -> id int not null primary key auto_increment,
    -> item_name varchar (50) not null,
    -> item_desc text,
    -> item_price float not null,
    -> curr_qty int not null
    -> );
Query OK, 0 rows affected (0.02 sec)
```

Apropo

> Câmpul `id` este definit drept cheie primară. Veți învăța mai multe despre chei în capitolele următoare, în contextul creării de tabele specifice în cadrul unor aplicații ilustrative. Folosind atributul `auto_increment` pentru un câmp, îi spuneți lui MySQL să adauge numărul imediat următor care este disponibil în câmpul `id` în locul dumneavoastră.

Serverul MySQL va răspunde cu mesajul `Query OK` (Interogare OK) de câte ori o comandă, indiferent de tipul acesteia, este executată cu succes. În caz contrar, va fi afișat un mesaj de eroare.

Folosirea comenzii `INSERT`

După ce creaţi câteva tabele, veţi folosi comanda `INSERT` din SQL pentru a adăuga înregistrări noi în aceste tabele. Sintaxa de bază a comenzii `INSERT` este:

```
INSERT INTO nume_tabela (lista coloane) VALORI (valori pentru coloane);
```

În cadrul listei de valori din a doua paranteză, trebuie să introduceţi şiruri în ghilimele. Pentru SQL, standardul este reprezentat de ghilimelele ('), dar MySQL vă permite să folosiţi şi ghilimele duble ("). Nu uitaţi că trebuie să folosiţi caracterul de eludare (\) pentru a evita procesarea ghilimelelor care se găsesc, eventual, în şirul efectiv.

Apropo

> Numerele întregi nu trebuie puse între semne de citare.

Iată un exemplu de şir de caractere în cazul căruia trebuie folosit caracterul de eludare:

```
O'Connor said "Boo"
```

Dacă aţi pune acest şir între ghilimele duble, comanda `INSERT` ar arăta astfel:

```
INSERT INTO table_name (column_name) VALUES ("O'Connor said \"Boo\"");
```

Dacă puneţi acest şir în ghilimele simple, însă, comanda `INSERT` va trebui să arate astfel:

```
INSERT INTO table_name (column_name) VALUES ('O\'Connor said "Boo"');
```

Să privim mai îndeaproape comanda `INSERT`

Pe lângă numele tabelei, există încă două părţi mari ale unei instrucţiuni `INSERT`: lista de coloane şi lista de valori. Numai lista de valori este necesară cu adevărat, dar dacă omiteţi lista de coloane, trebuie să prevedeţi în mod specific şi în ordine fiecare coloană în cadrul listei de valori.

Folosind drept exemplu tabela `grocery_inventory`, avem cinci câmpuri: `id`, `item_name`, `item_desc`, `item_price` şi `curr_qty`. Pentru a insera o înregistrare completă, aţi putea folosi oricare dintre instrucţiunile care urmează:

- o instrucţiune în care sunt menţionate toate coloanele:

```
insert into grocery_inventory (id, item_name, item_desc,
item_price, curr_qty)
values ('1', 'Apples', 'Beautiful, ripe apples.', '0.25', 1000);
```

- o instrucţiune care foloseşte toate coloanele dar nu le menţionează explicit:

```
insert into grocery_inventory values ('2', 'Bunches of Grapes',
'Seedless grapes.', '2.99', 500);
```

Încercaţi ambele variante şi vedeţi ce se întâmplă. Ar trebui să obţineţi rezultate asemănătoare cu cele de mai jos:

```
mysql> insert into grocery_inventory
    -> (id, item_name, item_desc, item_price, curr_qty)
    -> values
    -> (1, 'Apples', 'Beautiful, ripe apples.', 0.25, 1000);
Query OK, 1 row affected (0.01 sec)

mysql> insert into grocery_inventory values (2, 'Bunches of Grapes',
    -> 'Seedless grapes.', 2.99, 500);
Query OK, 1 row affected (0.01 sec)
```

Acum, haideţi să vedem câteva metode mai interesante de folosire a comenzii `INSERT`. Deoarece `id` este un număr întreg care se incrementează automat, nu trebuie să îl puneţi în lista de valori. Totuşi, dacă vreţi să nu introduceţi o anumită valoare (cum ar fi cea pentru câmpul `id`), atunci trebuie să enumeraţi tot restul coloanelor pentru care vreţi să introduceţi valori. De exemplu, următoare comandă nu enumeră coloanele şi nici nu introduce o valoare pentru `id`, aşa că va provoca o eroare:

```
mysql> insert into grocery_inventory values
    -> ('Bottled Water (6-pack)', '500ml spring water.', 2.29, 250);
ERROR 1136: Column count doesn't match value count at row 1
```

Din cauză că nu aţi enumerat nici o coloană, MySQL se aşteaptă să existe valori pentru toate în lista de valori, fapt ce provoacă o eroare la introducerea comenzii precedente. Dacă scopul era să îl lăsaţi pe MySQL să lucreze în locul dumneavoastră auto-incrementând câmpul `id`, puteaţi folosi oricare dintre variantele de comandă de mai jos:

- o comandă care enumeră toate coloanele, cu excepţia coloanei `id`:

```
insert into grocery_inventory (item_name, item_desc, item_price,
curr_qty)
values ('Bottled Water (6-pack)', '500ml spring water.', '2.29', 250);
```

- o comandă care foloseşte toate coloanele, dar nu specifică explicit nici una şi indică o valoarea `NULL` pentru `id` (astfel încât MySQL să completeze în locul dumneavoastră valoarea):

```
insert into grocery_inventory values ('NULL', 'Bottled Water (6-pack)',
'500ml spring water.', 2.29, 250);
```

Alegeţi una dintre variante şi folosiţi-o astfel încât tabela grocery_inventory să aibă trei înregistrări în total. Nu contează pentru MySQL ce variantă folosiţi, dar, la fel ca în cazul oricărui lucru care se bazează pe preferinţele utilizatorului, ar trebui să fiţi consecvent atunci când dezvoltaţi o aplicaţie. Structurile consecvente sunt mai uşor de depanat mai târziu deoarece veţi şti la ce să vă aşteptaţi.

Folosirea comenzii SELECT

SELECT este comanda SQL pentru extragerea de înregistrări. Sintaxa acestei comenzi poate fi complet simplistă sau poate să fie extrem de complicată. Pe măsură ce vă veţi obişnui cu programarea bazelor de date, veţi învăţa să îmbunătăţiţi instrucţiunile SELECT, ajungând, până la urmă, să exploataţi la maxim posibilităţile bazei dumneavoastră de date fără a suprasolicita limbajul de programare.

Sintaxa cea mai rudimentară a unei instrucţiuni SELECT arată astfel:

```
SELECT expresii_si_coloane FROM nume_tabela
[WHERE o_conditie_este_adevarata]
[ORDER BY o_coloana_oarecare [ASC | DESC]]
[LIMIT deplasare, randuri]
```

Priviţi prima linie:

```
SELECT expresii_si_coloane FROM nume_tabela
```

O expresie deosebit de utilă este reprezentată de simbolul *, care înseamnă totul. Astfel, pentru a selecta totul (toate rândurile, toate coloanele) din tabela grocery_inventory, instrucţiunea SQL pe care trebuie să o folosiţi este:

```
SELECT * FROM grocery_inventory;
```

În funcţie de cât de multe date se găsesc în tabela grocery_inventory, rezultatele dumneavoastră vor fi diferite, dar vor arăta similar cu cele de mai jos:

```
mysql> select * from grocery_inventory;
+---+----------------------+----------------------+-----------+---------+
| id| item_name            | item_desc            | item_price| curr_qty|
+---+----------------------+----------------------+-----------+---------+
|  1| Apples               | Beautiful, ripe apples.|      0.25|     1000|
|  2| Bunches of Grapes    | Seedless grapes.     |      2.99|      500|
|  3| Bottled Water (6-pack)| 500ml spring water.  |      2.29|      250|
+---+----------------------+----------------------+-----------+---------+
3 rows in set (0.00 sec)
```

După cum puteţi vedea, MySQL creează un tabel frumos cu numele coloanelor pe primul rând în cadrul setului de rezultate. Dacă vrei să selectaţi numai anumite coloane, înlocuiţi * cu numele coloanelor, separate prin virgule.

Următoarea instrucţiune selectează numai câmpurile id, item_name şi curr_qty din tabela grocery_inventory:

```
mysql> select id, item_name, curr_qty from grocery_inventory;
+----+-----------------------+----------+
| id | item_name             | curr_qty |
+----+-----------------------+----------+
|  1 | Apples                |     1000 |
|  2 | Bunches of Grapes     |      500 |
|  3 | Bottled Water (6-pack) |      250 |
+----+-----------------------+----------+
3 rows in set (0.00 sec)
```

Ordonarea rezultatelor comenzii SELECT

Rezultatele interogărilor de tip SELECT sunt ordonate aşa cum au fost introduse în tabelă şi acesta nu poate fi un criteriu inteligibil de ordonare. Dacă vreţi să ordonaţi rezultatele într-un anume mod, cum ar fi în funcţie de dată, ID, nume şi aşa mai departe, trebuie să vă specificaţi dorinţa prin intermediul unei clauze ORDER BY. În următoarea instrucţiune, rezultatele sunt ordonate pe baza câmpului item_name:

```
mysql> select id, item_name, curr_qty from grocery_inventory
    -> order by item_name;
+----+-----------------------+----------+
| id | item_name             | curr_qty |
+----+-----------------------+----------+
|  1 | Apples                |     1000 |
|  3 | Bottled Water (6-pack) |      250 |
|  2 | Bunches of Grapes     |      500 |
+----+-----------------------+----------+
3 rows in set (0.04 sec)
```

Ştiaţi că...

Atunci când selectaţi rezultate dintr-o tabelă fără să specificaţi o ordine de sortare, rezultatele pot sau nu să fie sortate după valoarea cheii primare. Acest lucru se întâmplă deoarece MySQL reutilizează spaţiul ocupat de rândurile şterse. Cu alte cuvinte, dacă adăugaţi înregistrări cu valorile ID-ului de la 1 până la 5, ştergeţi înregistrarea cu ID-ul 4 şi, apoi, adăugaţi încă o înregistrare (cu ID 6), înregistrările ar putea apărea în tabelă în această ordine: 1, 2, 3, 6, 5.

Rezultatele sortate cu ORDER BY sunt sortate implicit în ordine crescătoare (ASC); şirurile de caractere sunt sortate de la A la Z, numerele întregi încep de la 0, datele de la cele mai vechi la cele mai noi. Puteţi, de asemenea, să specificaţi o ordine descrescătoare (DESC):

```
mysql> select id, item_name, curr_qty from grocery_inventory
    -> order by item_name desc;
```

```
+----+-------------------------+----------+
| id | item_name               | curr_qty |
+----+-------------------------+----------+
|  2 | Bunches of Grapes       |      500 |
|  3 | Bottled Water (6-pack)  |      250 |
|  1 | Apples                  |     1000 |
+----+-------------------------+----------+
3 rows in set (0.00 sec)
```

Nu sunteţi limitaţi la sortarea după un singur câmp — puteţi specifica oricâte câmpuri doriţi, separate prin virgule. Prioritatea de sortare este dată de ordinea în care aţi enumerat câmpurile.

Limitarea rezultatelor

Puteţi folosi clauza `LIMIT` pentru ca interogarea `SELECT` să returneze numai un anumit număr de înregistrări. Există două argumente obligatorii la folosirea clauzei `LIMIT`: deplasarea şi numărul de rânduri. Deplasarea reprezintă poziţia de start, iar numărul de rânduri... ghici!

Să presupunem că aţi avea mai mult de două sau trei înregistrări în tabela `grocery_inventory` şi că aţi vrea să selectaţi ID-ul, numele şi cantitatea primelor trei, ordonate după câmpul `curr_qty`. Cu alte cuvinte, doriţi să selectaţi cele mai slab reprezentate trei elemente din stoc. Următoarea limită cu un singur parametru va porni de la poziţia `0` şi va merge până la a treia înregistrare:

```
mysql> select id, item_name, curr_qty from grocery_inventory
    -> order by curr_qty limit 3;
+----+-------------------------+----------+
| id | item_name               | curr_qty |
+----+-------------------------+----------+
|  4 | Bananas                 |      150 |
|  3 | Bottled Water (6-pack)  |      250 |
|  2 | Bunches of Grapes       |      500 |
+----+-------------------------+----------+
3 rows in set (0.00 sec)
```

Clauza `LIMIT` poate fi extrem de utilă într-o aplicaţie reală. De exemplu, puteţi folosi `LIMIT` în cadrul unei serii de instrucţiuni `SELECT` pentru a parcurge rezultatele în etape:

```
1. SELECT * FROM grocery_inventory ORDER BY curr_qty LIMIT 0, 3;

2. SELECT * FROM grocery_inventory ORDER BY curr_qty LIMIT 3, 3;

3. SELECT * FROM grocery_inventory ORDER BY curr_qty LIMIT 6, 3;
```

Dacă specificaţi o deplasare şi un număr de rânduri în interogarea dumneavoastră pentru care nu este găsit nici un rezultat, nu va apărea nici un mesaj de eroare — doar un set de rezultate vid. De exemplu, dacă tabela `grocey_inventory` conţine numai şase înregistrări, o interogare cu o deplasare a clauzei `LIMIT` de 6 nu va avea nici un rezultat:

```
mysql> select id, item_name, curr_qty from grocery_inventory
    -> order by curr_qty limit 6, 3;
Empty set (0.00 sec)
```

În aplicaţiile pentru Web, atunci când datele sunt afişate cu linkuri de genul „precedentele 10" şi „următoarele 10", puteţi să puneţi pariu că a fost pusă la treabă clauza `LIMIT`.

Folosirea clauzei `WHERE` în interogările dumneavoastră

Aţi învăţat numeroase feluri de a extrage anumite coloane din tabelele dumneavoastră, dar nu şi cum să extrageţi anumite rânduri. Acesta este momentul în care intră în joc clauza `WHERE`. În exemplul de sintaxă a comenzii `SELECT` de mai jos, puteţi să observaţi că `WHERE` este folosită pentru a specifica o anumită condiţie:

```
SELECT expresii_si_coloane FROM nume_tabel
[WHERE o_conditie_este_adevarata]
```

Un exemplu ar fi extragerea tuturor înregistrărilor care au o cantitate de 500:

```
mysql> select * from grocery_inventory where curr_qty = 500;
+----+------------------+-----------------------+------------+----------+
| id | item_name        | item_desc             | item_price | curr_qty |
+----+------------------+-----------------------+------------+----------+
|  2 | Bunches of Grapes | Seedless grapes.      |       2.99 |      500 |
|  5 | Pears            | Anjou, nice and sweet. |        0.5 |      500 |
+----+------------------+-----------------------+------------+----------+
2 rows in set (0.00 sec)
```

După cum aţi văzut mai sus, dacă folosiţi un număr întreg în cadrul clauzei `WHERE`, ghilimelele nu sunt necesare. Ghilimelele sunt necesare în jurul şirurilor de caractere şi se aplică aceleaşi reguli cu privire la caracterele de eludare ca în cazul comenzii `INSERT`.

Folosirea operatorilor în clauzele `WHERE`

Aţi folosit semnul de egalitate (=) în clauzele dumneavoastră `WHERE` pentru a determina valoarea de adevăr a unei condiţii — dacă un lucru este egal cu un altul. Puteţi folosi multe tipuri de operatori, dintre care operatorii de comparare şi cei logici reprezintă tipurile cele mai populare (a se vedea tabelul 15.1).

TABELUL 15.1. Cei mai importanți operatori de comparare și semnificațiile fiecăruia.

Operator	Semnificație
=	Egal cu
!=	Diferit de
<=	Mai mic sau egal cu
<	Mai mic decât
>=	Mai mare sau egal cu
>	Mai mare decât

Mai există și un operator foarte la îndemână care se numește `BETWEEN` și care este folositor în cazul comparațiilor între numere întregi sau date calendaristice deoarece caută rezultate cuprinse între o valoare minimă și una maximă. De exemplu:

```
mysql> select * from grocery_inventory
    -> where item_price  between 1.50 and 3.00;
+----+-----------------------+--------------------+------------+----------+
| id | item_name             | item_desc          | item_price | curr_qty |
+----+-----------------------+--------------------+------------+----------+
|  2 | Bunches of Grapes     | Seedless grapes.   |       2.99 |      500 |
|  3 | Bottled Water (6-pack) | 500ml spring water. |       2.29 |      250 |
|  4 | Bananas               | Bunches, green.    |       1.99 |      150 |
+----+-----------------------+--------------------+------------+----------+
3 rows in set (0.00 sec)
```

Alți operatori ar mai fi operatorii logici, care vă permit să efectuați comparații multiple în cadrul aceleiași clauze `WHERE`. Operatorii logici de bază sunt `AND` și `OR`. Atunci când folosiți `AND`, toate comparațiile din cadrul clauzei trebuie să fie adevărate pentru ca rezultatul să fie extras, pe când, atunci când folosiți `OR`, dacă minim una dintre comparații este adevărată, rezultatul va fi extras. De asemenea, puteți folosi operatorul `IN` pentru a specifica o listă de valori pe care doriți să le găsiți.

Comparațiile între șiruri de caractere cu ajutorul operatorului `LIKE`

Vi s-a prezentat modul în care puteți verifica identitatea dintre șiruri în cadrul unei clauze `WHERE` folosind = sau `!=`, dar mai există încă un operator util pentru clauza `WHERE`. El este folosit pentru comparațiile între șirurile de caractere și se numește `LIKE`. Acest operator folosește două caractere generice pentru stabilirea identității între porțiuni ale șirurilor:

- `%` — substituie mai multe caractere;
- `_` — substituie exact un caracter.

Dacă vreți să găsiți înregistrările din tabela `grocery_inventory` ale căror nume începe cu litera `A`, folosiți:

```
mysql> select * from grocery_inventory where item_name like 'A%';
+----+-----------+------------------------+------------+----------+
| id | item_name | item_desc              | item_price | curr_qty |
+----+-----------+------------------------+------------+----------+
|  1 | Apples    | Beautiful, ripe apples. |       0.25 |     1000 |
|  6 | Avocado   | Large Haas variety.    |       0.99 |      750 |
+----+-----------+------------------------+------------+----------+
```

Apropo

În afara cazului în care efectuați o comparație LIKE între șiruri binare, comparația nu depinde de capitalizare. Puteți forța o comparație care să depindă de capitalizare folosind cuvântul cheie BINARY.

Selectarea înregistrărilor din mai multe tabele

Nu trebuie să selectaţi dintr-un singur tabel în acelaşi timp. Dacă ar fi aşa, programarea aplicaţiilor ar fi un proces lung şi anevoios! Atunci când selectaţi din mai multe tabele în cadrul unei singure instrucţiuni SELECT, deja conectaţi laolaltă tabelele.

Să presupunem că aţi avea două tabele: fruits şi color (fructe şi culoare). Puteţi selecta toate rândurile din ambele tabele, folosind două instrucţiuni SELECT separate.

```
mysql> select * from fruit;
+----+-----------+
| id | fruitname |
+----+-----------+
|  1 | apple     |
|  2 | orange    |
|  3 | grape     |
|  4 | banana    |
+----+-----------+
4 rows in set (0.00 sec)

mysql> select * from color;
+----+-----------+
| id | colorname |
+----+-----------+
|  1 | red       |
|  2 | orange    |
|  3 | purple    |
|  4 | yellow    |
+----+-----------+
4 rows in set (0.00 sec)
```

Atunci când doriţi să selectaţi din ambele tabele în acelaşi timp, apar câteva diferenţe în sintaxa instrucţiunii SELECT. Mai întâi, trebuie să vă asiguraţi că toate tabelele pe care le folosiţi în interogarea dumneavoastră apar în clauza FROM a instrucţiunii SELECT.

Folosind exemplul cu `fruits` şi `color`, dacă vreţi doar să selectaţi toate coloanele şi rândurile din ambele tabele, aţi putea crede că trebuie să folosiţi următoarea instrucţiune `SELECT`:

```
mysql> select * from fruit, color;
+----+-----------+----+-----------+
| id | fruitname | id | colorname |
+----+-----------+----+-----------+
|  1 | apple     |  1 | red       |
|  2 | orange    |  1 | red       |
|  3 | grape     |  1 | red       |
|  4 | banana    |  1 | red       |
|  1 | apple     |  2 | orange    |
|  2 | orange    |  2 | orange    |
|  3 | grape     |  2 | orange    |
|  4 | banana    |  2 | orange    |
|  1 | apple     |  3 | purple    |
|  2 | orange    |  3 | purple    |
|  3 | grape     |  3 | purple    |
|  4 | banana    |  3 | purple    |
|  1 | apple     |  4 | yellow    |
|  2 | orange    |  4 | yellow    |
|  3 | grape     |  4 | yellow    |
|  4 | banana    |  4 | yellow    |
+----+-----------+----+-----------+
16 rows in set (0.00 sec)
```

Probabil că nu doreaţi şaisprezece rânduri cu informaţii redundante! Ceea ce a făcut această interogare a fost să unească fiecare rând din tabela `color` cu fiecare rând din tabela `fruits`. Din moment ce avem patru înregistrări în tabela `fruits` şi patru înregistrări în tabela `color`, interogarea v-a returnat 16 rezultate.

Atunci când selectaţi din mai multe tabele, trebuie să construiţi clauze `WHERE` adecvate pentru a vă asigura că obţineţi cu adevărat ceea ce doriţi. În cazul tabelelor `fruit` şi `color`, ceea ce doriţi este să vedeţi înregistrările `fruitname` şi `colorname` din aceste tabele acolo unde ID-urile fiecăruia sunt aceleaşi. Acest lucru ne aduce la următoarea nuanţă a interogărilor de acest tip — cum să indicaţi exact ce câmpuri doriţi atunci când aceste câmpuri au acelaşi nume în ambele tabele.

Pur şi simplu adăugaţi numele tabelului la numele câmpului, în felul acesta:

```
numetabel.numecamp
```

Astfel, interogarea care să selecteze câmpurile `fruitname` şi `colorname` din ambele tabele acolo unde ID-urile se potrivesc devine:

```
mysql> select fruitname, colorname from fruit, color where fruit.id = color.id;
+-----------+-----------+
| fruitname | colorname |
+-----------+-----------+
| apple     | red       |
| orange    | orange    |
| grape     | purple    |
| banana    | yellow    |
+-----------+-----------+
4 rows in set (0.00 sec)
```

Cu toate acestea, dacă încercaţi să selectaţi o coloană care apare în ambele tabele şi are acelaşi nume, veţi provoca o eroare de ambiguitate:

```
mysql> select id, fruitname, colorname from fruit, color
    -> where fruit.id = color.id;
ERROR 1052: Column: 'id' in field list is ambiguous
```

Dacă doreaţi să selectaţi ID-ul din tabela `fruit`, ar fi trebuit să folosiţi:

```
mysql> select fruit.id, fruitname, colorname from fruit,
    -> color where fruit.id = color.id;
+------+-----------+-----------+
| id   | fruitname | colorname |
+------+-----------+-----------+
|    1 | apple     | red       |
|    2 | orange    | orange    |
|    3 | grape     | purple    |
|    4 | banana    | yellow    |
+------+-----------+-----------+
4 rows in set (0.00 sec)
```

Acesta a fost un exemplu simplificator de joncţiune a două tabele în cadrul unei interogări de tip `SELECT`. Cuvântul cheie `JOIN` (joncţiune) este parte integrantă din SQL şi vă permite să construiţi interogări mai complexe.

Folosirea cuvântului cheie `JOIN`

În MySQL pot fi folosite mai multe tipuri de joncţiuni cu ajutorul cuvântului cheie `JOIN` şi toate se referă la ordinea în care tabelele sunt puse împreună şi în care sunt afişate rezultatele. Tipurile de joncţiune folosite cu tabelele `fruit` şi `color` se numeşte joncţiune internă — `INNER JOIN` — deşi acest lucru nu a fost specificat ca atare în interogare. Pentru a rescrie instrucţiunea SQL utilizând sintaxa `INNER JOIN` adecvată, va trebui să folosiţi:

```
mysql> select fruitname, colorname from fruit
    -> inner join color on fruit.id = color.id;
+-----------+-----------+
| fruitname | colorname |
+-----------+-----------+
| apple     | red       |
| orange    | orange    |
| grape     | purple    |
| banana    | yellow    |
+-----------+-----------+
4 rows in set (0.00 sec)
```

Clauza ON înlocuieşte clauza WHERE, iar, în acest exemplu, îi spune lui MySQL să conecteze rândurile din tabele în care ID-urile se potrivesc unul cu celălalt. Atunci când creaţi joncţiuni între tabele cu ajutorul clauzelor ON, puteţi folosi toate expresiile condiţionale pe care le puteţi folosi într-o clauză WHERE, inclusiv toţi operatorii aritmetici şi logici.

Un alt tip frecvent utilizat de joncţiune este joncţiunea la stânga — LEFT JOIN. Atunci folosiţi LEFT JOIN, toate rândurile din prima tabelă vor fi returnate, indiferent dacă există sau nu potriviri cu cel de-a doua tabelă. Să presupunem că aveţi două tabele într-un repertoar de adrese, una denumită master_name conţinând înregistrările esenţiale şi alta denumită email, conţinând eventualele înregistrări cu adresele de e-mail. Oricare dintre înregistrările din tabelul email vor fi conectate cu un anume ID al unei înregistrări din tabela master_name. De exemplu:

```
mysql> select name_id, firstname, lastname from master_name;
+---------+-----------+----------+
| name_id | firstname | lastname |
+---------+-----------+----------+
|       1 | John      | Smith    |
|       2 | Jane      | Smith    |
|       3 | Jimbo     | Jones    |
|       4 | Andy      | Smith    |
|       7 | Chris     | Jones    |
|      45 | Anna      | Bell     |
|      44 | Jimmy     | Carr     |
|      43 | Albert    | Smith    |
|      42 | John      | Doe      |
+---------+-----------+----------+
9 rows in set (0.00 sec)

mysql> select name_id, email from email;
+---------+------------------+
| name_id | email            |
+---------+------------------+
|      42 | jdoe@yahoo.com   |
|      45 | annabell@aol.com |
+---------+------------------+
2 rows in set (0.00 sec)
```

Folosind LEFT JOIN asupra acestor două tabele, puteţi vedea că, dacă o valoarea din tabelul email nu există, în locul adresei de e-mail va apărea NULL.

```
mysql> select firstname, lastname, email fom master_name
    -> left join email on master_name.name_id = email.name_id;
+-----------+----------+------------------+
| firstname | lastname | email            |
+-----------+----------+------------------+
| John      | Smith    | NULL             |
| Jane      | Smith    | NULL             |
| Jimbo     | Jones    | NULL             |
| Andy      | Smith    | NULL             |
| Chris     | Jones    | NULL             |
| Anna      | Bell     | annabell@aol.com |
| Jimmy     | Carr     | NULL             |
| Albert    | Smith    | NULL             |
| John      | Doe      | jdoe@yahoo.com   |
+-----------+----------+------------------+
9 rows in set (0.01 sec)
```

O joncţiune la dreapta — `RIGHT JOIN` — funcţionează exact la fel ca `LEFT JOIN`, doar că ordinea tabelelor este inversată. Cu alte cuvinte, atunci când folosiţi `RIGHT JOIN`, toate rândurile din cea de-a doua tabelă vor fi returnate, indiferent dacă există potriviri sau nu cu înregistrări din prima tabelă. Totuşi, în cazul tabelelor `master_name` şi `email`, în tabela `email` nu există decât două înregistrări, pe când, în tabela `master_name`, se găsesc nouă înregistrări. Acest lucru înseamnă că vor fi returnate doar două înregistrări din cele nouă:

```
mysql> select firstname, lastname, email from master_name
    -> right join email on master_name.name_id = email.name_id;
+-----------+----------+------------------+
| firstname | lastname | email            |
+-----------+----------+------------------+
| John      | Doe      | jdoe@yahoo.com   |
| Anna      | Bell     | annabell@aol.com |
+-----------+----------+------------------+
2 rows in set (0.00 sec)
```

În MySQL mai sunt disponibile încă vreo câteva tipuri de `JOIN`, dar aţi învăţat despre cele mai folosite dintre acestea. Pentru a învăţa despre joncţiuni precum joncţiunea încrucişată (`CROSS JOIN`), directă (`STRAIGHT JOIN`) şi naturală (`NATURAL JOIN`), vă rog să vizitaţi Manualul de MySQL la adresa `http://www.mysql.com/doc/J/O/JOIN.html`.

Folosirea comenzii `UPDATE` pentru modificarea înregistrărilor

`UPDATE` (actualizare) este comanda SQL folosită pentru modificarea conţinutului uneia sau mai multor coloane dintr-o înregistrare existentă. Sintaxa elementară a comenzii `UPDATE` arată astfel:

```
UPDATE nume_tabel
SET coloana1='valoare noua 1',
coloana2='valoare noua 2'
[WHERE o_conditie_e_adevarata]
```

Regulile privind actualizarea unei înregistrări sunt similare cu cele folosite atunci când se inserează o înregistrare: Datele pe care le introduceţi trebuie să fie corespunzătoare tipului de date al câmpului şi trebuie să puneţi şirurile între ghilimele simple sau duble, folosind caractere de eludare acolo unde este necesar.

De exemplu, să presupunem că aveţi o tabelă denumită `fruit` conţinând un ID, un nume de fruct şi starea fructului (`ripe` — copt sau `rotten` — putred).

```
mysql> SELECT * FROM fruit;
+----+------------+--------+
| id | fruit_name | status |
+----+------------+--------+
|  1 | apple      | ripe   |
|  2 | pear       | rotten |
|  3 | banana     | ripe   |
|  4 | grape      | rotten |
+----+------------+--------+
4 rows in set (0.00 sec)
```

Pentru a actualiza starea fructului la `ripe`, trebuie să folosiţi:

```
mysql> update fruit set status = 'ripe';
Query OK, 2 rows affected (0.00 sec)
Rows matched: 4  Changed: 2  Warnings: 0
```

Priviţi rezultatul interogării dumneavoastră. A fost efectuată cu succes, după cum puteţi deduce din mesajul `Query OK`. De asemenea, remarcaţi că numai două rânduri au fost afectate — dacă încercaţi să modificaţi valoarea unei coloane la o valoare pe care aceasta o are deja, actualizarea nu va avea loc pentru respectiva coloană.

Cea de-a doua linie a răspunsului arată că au fost găsite patru rânduri, dintre care doar două au fost schimbate. Dacă vă întrebaţi ce rânduri au fost găsite, răspunsul este simplu: Din moment ce nu aţi specificat nici o condiţie care să trebuiască să fie îndeplinită, au fost găsite:

```
toate randurile
```

Trebuie să fiţi foarte atenţi când folosiţi o condiţie pentru a actualiza o tabelă, cu excepţia cazului în care vreţi să modificaţi toate coloanele tuturor înregistrărilor la aceeaşi valoare. Pentru ilustrare, să presupunem că, în tabelă, `"grape"` (strugure) ar fi scris incorect şi că doriţi să folosiţi `UPDATE` pentru a corecta această greşeală. Interogarea de mai jos ar avea rezultate dezastruoase:

```
mysql> update fruit set fruit_name = 'grape';
Query OK, 4 rows affected (0.00 sec)
Rows matched: 4  Changed: 4  Warnings: 0
```

Când aţi citi rezultatul, v-ar cuprinde panica: vor fi fost schimbate 4 din 4 înregistrări, ceea ce ar însemna că tabela dumneavoastră `fruit` arată acum astfel:

```
mysql> SELECT * FROM fruit;
+----+------------+--------+
| id | fruit_name | status |
+----+------------+--------+
|  1 | grape      | ripe   |
|  2 | grape      | ripe   |
|  3 | grape      | ripe   |
|  4 | grape      | ripe   |
+----+------------+--------+
4 rows in set (0.00 sec)
```

Toate fructele din înregistrări sunt acum struguri. În timp ce încercaţi să corectaţi un singur câmp, toate câmpurile au fost schimbate pentru că dumneavoastră nu aţi specificat nici o condiţie! Când le împărţiţi utilizatorilor dumneavoastră privilegii pentru comanda UPDATE, gândiţi-vă cu ce responsabilitate îi investiţi — o singură mişcare greşită şi toate tabelele pot să se umple cu struguri!

Instrucţiuni UPDATE condiţionale

Pentru a crea o instrucţiune UPDATE condiţională trebuie să folosiţi clauze WHERE pentru a fi găsite anumite înregistrări. Folosirea unei clauze WHERE într-o instrucţiune UPDATE este similară folosirii clauzelor WHERE în cadrul instrucţiunilor SELECT. Puteţi apela la toţi operatorii de comparare, precum şi la operatorii logici, cum ar fi =, >, OR şi AND.

Să presupunem că tabela dumneavoastră fruit nu a fost umplută complet cu struguri, şi că ea conţine patru înregistrări, dintre care una are o greşeală de scriere (grappe în loc de grape). Instrucţiunea UPDATE care ar rezolva problema ar fi:

```
mysql> update fruit set fruit_name = 'grape' where fruit_name = 'grappe';
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

În acest caz, numai un singur rând a fost găsit de interogare şi, deci, numai acel rând a fost modificat. Tabela dumneavoastră fruit ar trebui să fie intactă, atâta doar că toate numele fructelor sunt acum scrise corect:

```
mysql> select * from fruit;
+----+------------+--------+
| id | fruit_name | status |
+----+------------+--------+
|  1 | apple      | ripe   |
|  2 | pear       | ripe   |
|  3 | banana     | ripe   |
|  4 | grape      | ripe   |
+----+------------+--------+
4 rows in set (0.00 sec)
```

Folosirea valorilor existente în coloane în cadrul instrucțiunilor UPDATE

O altă caracteristică a comenzii UPDATE este capacitatea ei de a folosi valori care există deja într-o înregistrare pe post de valoare de bază. De exemplu, întorcându-ne la tabela grocery_inventory pe care am folosit-o mai devreme în acest capitol:

```
mysql> select * from grocery_inventory;
+----+----------------------+-----------------------+------------+----------+
| id | item_name            | item_desc             | item_price | curr_qty |
+----+----------------------+-----------------------+------------+----------+
|  1 | Apples               | Beautiful, ripe apples.|       Ø.25 |     1ØØØ |
|  2 | Bunches of Grapes    | Seedless grapes.      |       2.99 |      5ØØ |
|  3 | Bottled Water (6-pack)| 5ØØml spring water.  |       2.29 |      25Ø |
|  4 | Bananas              | Bunches, green.       |       1.99 |      15Ø |
|  5 | Pears                | Anjou, nice and sweet.|        Ø.5 |      5ØØ |
|  6 | Avocado              | Large Haas variety.   |       Ø.99 |      75Ø |
+----+----------------------+-----------------------+------------+----------+
6 rows in set (Ø.ØØ sec)
```

Atunci când cineva cumpără un măr, tabela cu inventarul trebuie să fie actualizată corespunzător. Cu toate acestea, nu ştiţi cu exactitate ce număr să introduceţi în coloana curr_qty, ştiţi doar că un măr a fost vândut. În acest caz, puteţi folosi valoarea curentă din coloană şi să o decrementaţi cu o unitate:

```
mysql> update grocery_inventory set curr_qty = curr_qty - 1 where id = 1;
Query OK, 1 row affected (Ø.ØØ sec)
Rows matched: 1  Changed: 1  Warnings: Ø
```

În urma acestei comenzi, valoarea din curr_qty ar trebui să fie 999 şi, într-adevăr, aşa stau lucrurile:

```
mysql> select * from grocery_inventory;
+----+----------------------+-----------------------+------------+----------+
| id | item_name            | item_desc             | item_price | curr_qty |
+----+----------------------+-----------------------+------------+----------+
|  1 | Apples               | Beautiful, ripe apples.|       Ø.25 |      999 |
|  2 | Bunches of Grapes    | Seedless grapes.      |       2.99 |      5ØØ |
|  3 | Bottled Water (6-pack)| 5ØØml spring water.  |       2.29 |      25Ø |
|  4 | Bananas              | Bunches, green.       |       1.99 |      15Ø |
|  5 | Pears                | Anjou, nice and sweet.|        Ø.5 |      5ØØ |
|  6 | Avocado              | Large Haas variety.   |       Ø.99 |      75Ø |
+----+----------------------+-----------------------+------------+----------+
6 rows in set (Ø.ØØ sec)
```

Folosirea comenzii REPLACE

O altă metodă de modificare a înregistrărilor este prin folosirea comenzii REPLACE, care este extrem de similară comenzii INSERT.

```
REPLACE INTO nume_tabel (lista coloane) VALUES (valori pentru coloane);
```

Instrucţiunea REPLACE funcţionează astfel: dacă înregistrarea pe care o inseraţi în tabel conţine o valoare de cheie primară care se potriveşte cu o înregistrare care există deja în tabel, atunci înregistrarea din tabel va fi ştearsă, iar noua înregistrare va fi inserată în locul ei.

Apropo

Comanda REPLACE este o extensie adusă limbajului ANSI SQL şi este specifică pentru MySQL. Această comandă imită acţiunea secvenţei de instrucţiuni DELETE şi INSERT. Cu alte cuvinte, aveţi la dispoziţie două comenzi la preţul uneia singure.

Folosind tabela gorcery_inventory, următoarea comandă va înlocui intrarea corespunzătoare merelor:

```
mysql> replace into grocery_inventory values
    -> (1, 'Granny Smith Apples', 'Sweet!', '0.50', 1000);
Query OK, 2 rows affected (0.00 sec)
```

În rezultatul interogării, observaţi că se afirmă 2 rows affected (2 rânduri au fost afectate). În acest caz, din cauză că id este o cheie primară care se regăsea în tabela grocery_inventory, rândul original a fost şters şi a fost introdus un nou rând, deci au fost afectate două rânduri.

Selectaţi înregistrările pentru a verifica corectitudinea rezultatelor, adică:

```
mysql> select * from grocery_inventory;
+----+----------------------+----------------------+------------+----------+
| id | item_name            | item_desc            | item_price | curr_qty |
+----+----------------------+----------------------+------------+----------+
|  1 | Granny Smith Apples  | Sweet!               |        0.5 |     1000 |
|  2 | Bunches of Grapes    | Seedless grapes.     |       2.99 |      500 |
|  3 | Bottled Water (6-pack)| 500ml spring water. |       2.29 |      250 |
|  4 | Bananas              | Bunches, green.      |       1.99 |      150 |
|  5 | Pears                | Anjou, nice and sweet.|      0.5 |      500 |
|  6 | Avocado              | Large Haas variety.  |       0.99 |      750 |
+----+----------------------+----------------------+------------+----------+
6 rows in set (0.00 sec)
```

Dacă folosiţi o instrucţiune REPLACE, iar valoarea cheii primare din înregistrarea nouă nu se potriveşte cu valoarea cheii primare a nici unei înregistrări existente în tabelă, atunci înregistrarea va fi pur şi simplu inserată şi numai un singur rând va fi afectat.

Folosirea comenzii DELETE

Sintaxa de bază a comenzii DELETE este:

```
DELETE FROM nume_tabel
[WHERE o_conditie_e_adevarata]
[LIMIT randuri]
```

Observaţi că, în comanda DELETE, nu este specificată în nici un fel coloana — atunci când folosiţi DELETE, este eliminată întreaga înregistrare. Poate vă amintiţi fiascoul petrecut mai sus, în problema strugurilor din tabelul fruit, când actualizarea unei tabele fără specificarea unei condiţii a dus la actualizarea tuturor înregistrărilor. Trebuie să fiţi tot atât de precauţi atunci când folosiţi comanda DELETE.

Presupunând că structura şi datele din tabela numită fruit sunt ca mai jos:

```
mysql> select * from fruit;
+----+------------+--------+
| id | fruit_name | status |
+----+------------+--------+
|  1 | apple      | ripe   |
|  2 | pear       | rotten |
|  3 | banana     | ripe   |
|  4 | grape      | rotten |
+----+------------+--------+
4 rows in set (0.00 sec)
```

Instrucţiunea următoare va şterge toate înregistrările din tabelă:

```
mysql> delete from fruit;
Query OK, 0 rows affected (0.00 sec)
```

Toate fructele au dispărut.

Ştergerea condiţională

Adăugarea unei condiţii la comanda DELETE se face la fel ca în cazul comenzilor SELECT şi UPDATE, adică prin folosirea clauzelor WHERE pentru a fi găsite anumite înregistrări. Aveţi la dispoziţie întreaga gamă de comparaţii şi operatori logici, astfel încât să puteţi alege înregistrările pe care doriţi să le ştergeţi.

Un prim exemplu ar fi reprezentat de eliminarea tuturor înregistrărilor corespunzătoare fructelor putrede din tabela fruit:

```
mysql> delete from fruit where status = 'rotten';
Query OK, 2 rows affected (0.00 sec)
```

Au fost şterse două înregistrări, iar acum au rămas numai fructele coapte.

```
mysql> select * from fruit;
+----+------------+--------+
| id | fruit_name | status |
+----+------------+--------+
|  1 | apple      | ripe   |
|  3 | banana     | ripe   |
+----+------------+--------+
2 rows in set (0.00 sec)
```

Dacă folosiţi MySQL 4.0 (sau o versiune mai recentă), puteţi utiliza şi clauze `ORDER BY` în cadrul instrucţiunilor `DELETE`. Priviţi sintaxa de bază a instrucţiunii `DELETE` cu clauza `ORDER BY` adăugată în structura sa:

```
DELETE FROM nume_tabel
[WHERE o_conditie_e_adevarata]
[ORDER BY o_coloana [ASC | DESC]]
[LIMIT randuri]
```

Prima reacţie ar fi să vă întrebaţi: „Ce contează în ce ordine sunt şterse înregistrările?" Clauza `ORDER BY` nu specifică ordinea de ştergere a înregistrărilor, ci ordinea de sortare a acestora.

În exemplul de mai jos, un tabel denumit `access_log` (jurnal de acces) arată data şi ora la care a avut loc accesul şi numele de utilizator.

```
mysql> select * from access_log;
+----+---------------------+----------+
| id | date_accessed       | username |
+----+---------------------+----------+
|  1 | 2001-11-06 06:09:13 | johndoe  |
|  2 | 2001-11-06 06:09:22 | janedoe  |
|  3 | 2001-11-06 06:09:39 | jsmith   |
|  4 | 2001-11-06 06:09:44 | mikew    |
+----+---------------------+----------+
4 rows in set (0.00 sec)
```

Pentru a elimina cea mai veche înregistrare, trebuie să folosiţi mai întâi clauza `ORDER BY` pentru a sorta rezultatele corespunzător, iar, apoi, clauza `LIMIT` pentru a elimina o singură înregistrare:

```
mysql> delete from access_log order by date_accessed desc limit 1;
Query OK, 1 row affected (0.01 sec)
```

Selectaţi înregistrările din tabela `access_log` pentru a verifica dacă acum există doar trei înregistrări:

```
mysql> select * from access_log;
+----+---------------------+----------+
| id | date_accessed       | username |
+----+---------------------+----------+
|  2 | 2001-11-06 06:09:22 | janedoe  |
|  3 | 2001-11-06 06:09:39 | jsmith   |
|  4 | 2001-11-06 06:09:44 | mikew    |
+----+---------------------+----------+
3 rows in set (0.00 sec)
```

Funcţii de manipulare a şirurilor de caractere folosite în mod uzual în MySQL

Funcţiile dedicate şirurilor implementate în MySQL pot fi folosite în mai multe feluri. Puteţi să folosiţi o funcţie în cadrul instrucţiunilor SELECT fără a specifica o tabelă pentru a obţine rezultatul funcţiei respective. Alternativ, puteţi folosi funcţiile pentru a amplifica puterea instrucţiunilor dumneavoastră SELECT, de exemplu prin concatenarea a două câmpuri într-un singur şir de caractere nou.

Funcţii referitoare la lungime şi funcţii de concatenare

Grupul de funcţii referitoare la lungime şi de concatenare se concentrează asupra lungimii şirurilor de caractere şi asupra concatenării şirurilor de caractere. Printre funcţiile care operează cu lungimea şirurilor se află LENGTH(), OCTET_LENGTH(), CHAR_LENGTH() şi CHARACTER_LENGTH(), iar toate fac, în esenţă, acelaşi lucru: numără caracterele dintr-un şir de caractere.

```
mysql> select length('This is cool!');
+-------------------------+
| LENGTH('This is cool!') |
+-------------------------+
|                      13 |
+-------------------------+
1 row in set (0.00 sec)
```

Distracţia începe când ajungem la funcţia CONCAT(), funcţie ce este folosită pentru concatenarea a două sau mai multe şiruri de caractere:

```
mysql> select concat('My', 'S', 'QL');
+-------------------------+
| CONCAT('My', 'S', 'QL') |
+-------------------------+
| MySQL                   |
+-------------------------+
1 row in set (0.00 sec)
```

Imaginaţi-vă că folosiţi această funcţie asupra unei tabele care conţine nume fragmentate în firstname (prenume) şi lastname (nume). În loc să folosiţi două şiruri de caractere, puteţi folosi cele două câmpuri pentru a concatena prenumele cu numele. Concatenând aceste două câmpuri, puteţi reduce liniile de cod necesare pentru obţinerea unui aceluiaşi rezultat în aplicaţiile dumneavoastră:

```
mysql> select concat(firstname, lastname) from table_name;
+----------------------------+
| CONCAT(firstname, lastname) |
+----------------------------+
| JohnSmith                  |
| JaneSmith                  |
| JimboJones                 |
| AndySmith                  |
```

```
| ChrisJones                |
| AnnaBell                  |
| JimmyCarr                 |
| AlbertSmith               |
| JohnDoe                   |
+---------------------------+
9 rows in set (Ø.ØØ sec)
```

Dacă folosiți un nume de câmp şi nu un şir în cadrul unei funcții, nu includeți numele câmpului între ghilimele. Dacă faceți acest lucru, MySQL va interpreta şirul în mod literal. În exemplul de folosire a funcției `CONCAT()`, ați obține rezultatele următoare:

Ştiaţi că...

```
mysql> select concat('firstname', 'lastname') FROM table_name;
+---------------------------------+
| CONCAT('firstname', 'lastname') |
+---------------------------------+
| firstnamelastname               |
| firstnamelastname               |
| firstnamelastname               |
| firstnamelastname               |
| firstnamelastname               |
| firstnamelastname               |
| firstnamelastname               |
| firstnamelastname               |
| firstnamelastname               |
+---------------------------------+
9 rows in set (Ø.ØØ sec)
```

Funcţia `CONCAT()` ar putea fi utilă dacă ar exista un şi soi de separator între nume şi prenume, acesta fiind, însă efectul funcţiei `CONCAT_WS()`.

După cum, probabil, v-aţi dat seama, `CONCAT_WS()` provine de la *concatenate with separator* (*concatenare cu separator*). Separatorul poate fi orice vreţi dumneavoastră, dar exemplul de mai jos utilizează un spaţiu alb.

```
mysql> select concat_ws(' ', firstname, lastname) FROM table_name;
+-----------------------------------+
| CONCAT_WS(' ', firstname, lastname) |
+-----------------------------------+
| John Smith                        |
| Jane Smith                        |
| Jimbo Jones                       |
| Andy Smith                        |
| Chris Jones                       |
| Anna Bell                         |
| Jimmy Carr                        |
| Albert Smith                      |
| John Doe                          |
+-----------------------------------+
9 rows in set (Ø.ØØ sec)
```

Dacă doriţi să micşoraţi lăţimea tabelului dumneavoastră de rezultate, puteţi folosi clauza AS pentru a da câmpului de rezultate un nume definit de utilizator.

```
mysql> select concat_ws(' ', firstname, lastname) AS fullname FROM table_name;
+--------------+
| fullname     |
+--------------+
| John Smith   |
| Jane Smith   |
| Jimbo Jones  |
| Andy Smith   |
| Chris Jones  |
| Anna Bell    |
| Jimmy Carr   |
| Albert Smith |
| John Doe     |
+--------------+
9 rows in set (0.00 sec)
```

Funcţii de ajustare şi completare

MySQL ne pune la dispoziţie funcţii pentru adăugarea şi eliminarea caracterelor suplimentare (inclusiv a spaţiilor albe) din şirurile de caractere. Funcţiile RTRIM() şi LTRIM() elimină spaţiile albe de la sfârşitul şi, respectiv, începutul şirurilor de caractere:

```
mysql> select rtrim('stringstring   ');
+--------------------------+
| RTRIM('stringstring   ') |
+--------------------------+
| stringstring             |
+--------------------------+
1 row in set (0.00 sec)

mysql> select ltrim('  stringstring');
+--------------------------+
| LTRIM('  stringstring') |
+--------------------------+
| stringstring             |
+--------------------------+
1 row in set (0.00 sec)
```

Aţi putea avea nevoie să curăţaţi un şir de caractere dacă acesta provine dintr-un câmp cu lungime fixă şi fie nu are nevoie să păstreze caracterele suplimentare, fie trebuie introdus într-un câmp de tip varchar sau alt tip de câmp fără lungime fixată. Dacă şirurile de caractere sunt completate cu alte caractere decât spaţiile albe, folosiţi funcţia TRIM() pentru a specifica acele caractere pe care doriţi să le eliminaţi. De exemplu, pentru a elimina X-urile din faţa şirului XXXneedleXXX, folosiţi clauza LEADING:

```
mysql> select trim(leading 'X' from 'XXXneedleXXX');
+-------------------------------------+
| TRIM(LEADING 'X' from 'XXXneedleXXX') |
+-------------------------------------+
| needleXXX                           |
+-------------------------------------+
1 row in set (Ø.ØØ sec)
```

Folosiţi clauza `TRAILING` pentru a elimina caracterele de la sfârşitul şirului:

```
mysql> select trim(trailing 'X' from 'XXXneedleXXX');
+--------------------------------------+
| TRIM(TRAILING 'X' from 'XXXneedleXXX') |
+--------------------------------------+
| XXXneedle                            |
+--------------------------------------+
1 row in set (Ø.ØØ sec)
```

Dacă nu se specifică nici `LEADING` şi nici `TRAILING`, MySQL acţionează ca şi când ambele ar fi prezente:

```
mysql> select trim('X' from 'XXXneedleXXX');
+-------------------------------+
| TRIM('X' from 'XXXneedleXXX') |
+-------------------------------+
| needle                        |
+-------------------------------+
1 row in set (Ø.ØØ sec)
```

Tot aşa cum funcţiile `RTRIM()` şi `LTRIM()` elimină caracterele de completare, `RPAD()` şi `LPAD()` adaugă astfel de caractere la un şir. De exemplu, dacă doriţi să adăugaţi caractere speciale de identificare la un şir care este parte din numărul unei comenzi, într-o bază de date folosită pentru vânzări. Atunci când folosiţi funcţii de completare, elementele obligatorii sunt şirul de caractere, lungimea ţintă şi caracterul cu care să se facă completarea. De exemplu, completăm şirul `needle` cu caracterul `X` până ce şirul ajunge la lungimea de 10 caractere:

```
mysql> select rpad('needle', 1Ø, 'X');
+-------------------------+
| RPAD('needle', 1Ø, 'X') |
+-------------------------+
| needleXXXX              |
+-------------------------+
1 row in set (Ø.ØØ sec)

mysql> select lpad('needle', 1Ø, 'X');
+-------------------------+
| LPAD('needle', 1Ø, 'X') |
+-------------------------+
| XXXXneedle              |
+-------------------------+
1 row in set (Ø.ØØ sec)
```

Funcţii de găsire şi poziţionare

Grupul funcţiilor de găsire şi poziţionare este folositor pentru găsirea unor subşiruri de caractere în interiorul altor şiruri de caractere. Funcţia `LOCATE()` returnează poziţia primei apariţii a unui subşir dat în interiorul şirului ţintă. De exemplu, puteţi căuta acul (`needle`) în carul cu fân (`haystack`):

```
mysql> select locate('needle', 'haystackneedlehaystack');
+-----------------------------------------+
| LOCATE('needle', 'haystackneedlehaystack') |
+-----------------------------------------+
|                                       9 |
+-----------------------------------------+
1 row in set (0.00 sec)
```

Subşirul `needle` începe la poziţia 9 în interiorul şirului ţintă. Dacă subşirul nu poate fi găsit în interiorul şirului ţintă, MySQL returnează valoarea 0 drept rezultat.

Apropo

> Spre deosebire de funcţiile de care operează cu poziţia în cadrul şirurilor de caractere din majoritatea limbajelor de programare în cazul cărora numerotarea începe de la 0, poziţiile sunt numerotate, în MySQL începând de la 1.

O extensie a funcţiei `LOCATE()` este reprezentată de folosirea unui al treilea argument — poziţia de start. Dacă începeţi să căutaţi `needle` în `haystack` înainte de poziţia 9, veţi obţine un rezultat. Altminteri, din cauză că `needle` începe la poziţia 9, veţi obţine valoarea 0 dacă specificaţi o poziţie de start mai mare decât 9.

```
mysql> select locate('needle', 'haystackneedlehaystack',6);
+-------------------------------------------+
| LOCATE('needle', 'haystackneedlehaystack',9) |
+-------------------------------------------+
|                                         9 |
+-------------------------------------------+
1 row in set (0.00 sec)

mysql> select locate('needle', 'haystackneedlehaystack',12);
+-------------------------------------------+
| LOCATE('needle', 'haystackneedlehaystack',12) |
+-------------------------------------------+
|                                         0 |
+-------------------------------------------+
1 row in set (0.00 sec)
```

Funcţii care operează cu subşiruri

Dacă scopul dumneavoastră este să extrageţi un subşir dintr-un şir ţintă, puteţi utiliza câteva funcţii MySQL. Dacă aveţi un şir de caractere, o poziţie de start şi o lungime, puteţi folosi funcţia `SUBSTRING()`.

Acest exemplu preia trei caractere din şirul MySQL, începând de la poziţia 2:

```
mysql> select substring("MySQL", 2, 3);
+--------------------------+
| SUBSTRING("MySQL", 2, 3) |
+--------------------------+
| ySQ                      |
+--------------------------+
1 row in set (0.00 sec)
```

Dacă vreţi doar câteva caractere de la capătul stâng sau drept al unui şir de caractere, puteţi folosi funcţiile `LEFT()` şi `RIGHT()`.

```
mysql> select left("MySQL", 2);
+------------------+
| LEFT("MySQL", 2) |
+------------------+
| My               |
+------------------+
1 row in set (0.00 sec)

mysql> select right("MySQL", 3);
+-------------------+
| RIGHT("MySQL", 3) |
+-------------------+
| SQL               |
+-------------------+
1 row in set (0.00 sec)
```

Unul dintre cele mai uzuale scopuri în care sunt folosite funcţiile ce operează cu subşiruri este extragerea părţilor din numerele comenzilor pentru a afla cine a făcut o anumită comandă. În unele aplicaţii, sistemul este conceput să genereze automat un număr al comenzii care să conţină o dată calendaristică, un identificator al clientului şi alte informaţii. Dacă acest număr are întotdeauna o anumită structură, cum ar fi `XXXX-YYYYY-ZZ`, puteţi folosi funcţiile de subşiruri pentru a extrage componentele individuale din întregul număr. De exemplu, dacă `ZZ` reprezintă întotdeauna statul în care a fost expediată comanda, puteţi folosi funcţia `RIGHT()` pentru a extrage aceste caractere şi a raporta numărul de comenzi expediate într-un anumit stat.

Funcţii de modificare a şirurilor de caractere

Fără îndoială că limbajul dumneavoastră preferat de programare are funcţii care modifică aspectul şirurilor de caractere, dar, dacă puteţi să efectuaţi aceste operaţiuni în cadrul unor instrucţiuni SQL, este cu atât mai bine.

Funcţiile MySQL `LCASE()` şi `UCASE()` transformă un şir în litere mici sau litere mari:

```
mysql> select lcase('MYSQL');
+----------------+
| LCASE('MYSQL') |
+----------------+
| mysql          |
+----------------+
1 row in set (0.00 sec)

mysql> select ucase('mysql');
+----------------+
| UCASE('mysql') |
+----------------+
| MYSQL          |
+----------------+
1 row in set (0.00 sec)
```

Reţineţi că, dacă folosiţi funcţiile cu nume de câmpuri drept argumente, trebuie să nu folosiţi ghilimele.

```
mysql> select ucase(lastname) from table_name;
+-----------------+
| UCASE(lastname) |
+-----------------+
| BELL            |
| CARR            |
| DOE             |
| JONES           |
| JONES           |
| SMITH           |
| SMITH           |
| SMITH           |
| SMITH           |
+-----------------+
9 rows in set (0.00 sec)
```

O altă funcţie amuzantă de manipulare a şirurilor de caractere este `REPEAT()`, care face exact ce îi spune numele, adică repetă un şir de un număr dat de ori:

```
mysql> select repeat("bowwow", 4);
+--------------------------+
| REPEAT("bowwow", 4)      |
+--------------------------+
| bowwowbowwowbowwowbowwow |
+--------------------------+
1 row in set (0.00 sec)
```

Funcţia `REPLACE()` înlocuieşte toate apariţiile unui şir dat din interiorul unui şir ţintă:

```
mysql> select replace('bowwowbowwowbowwowbowwow', 'wow', 'WOW');
+-----------------------------------------------+
| REPLACE('bowwowbowwowbowwowbowwow', 'wow', 'WOW') |
+-----------------------------------------------+
| bowWOWbowWOWbowWOWbowWOW                      |
+-----------------------------------------------+
1 row in set (0.00 sec)
```

Folosirea funcțiilor MySQL dedicate datelor calendaristice şi orelor

Funcţiile dedicate datelor calendaristice şi implementate în MySQL pot fi folosite în cadrul instrucţiunilor SELECT, cu sau fără specificarea unei tabele, pentru a afla rezultatul unei funcţii. Alternativ, puteţi folosi funcţiile cu orice tip de câmp: `date`, `datetime`, `timestamp` sau `year`. În funcţie de tipul câmpului folosit, rezultatele funcţiilor calendaristice sunt mai mult sau mai puţin utile.

Operații cu zile

Funcţiile DAYOFWEEK() şi WEEKDAY() fac lucruri similare, dar returnează rezultate oarecum diferite. Ambele funcţii sunt folosite pentru a afla numărul de ordine în cadrul săptămânii pe care îl are o anumită dată calendaristică, dar diferenţa constă în ziua care este considerată începutul săptămânii.

Dacă folosiţi DAYOFWEEK(), prima zi a săptămânii este duminică, la poziţia 1, iar ultima zi a săptămânii este sâmbătă, la poziţia 7. De exemplu:

```
mysql> select dayofweek('2003-08-25');
+--------------------------+
| dayofweek('2003-08-25') |
+--------------------------+
|                        2 |
+--------------------------+
1 row in set (0.00 sec)
```

Rezultatele arată că data de 25 august 2003 a fost a 2-a zi din săptămână, sau luni. Folosind aceeaşi dată drept argument pentru funcţia WEEKDAY(), vom obţine alte rezultate:

```
mysql> select weekday('2003-08-25');
+-----------------------+
| WEEKDAY('2003-08-25') |
+-----------------------+
|                     0 |
+-----------------------+
1 row in set (0.00 sec)
```

Rezultatele arată acum că 25 august 2003 a fost ziua din săptămână cu numărul de ordine 0. Din moment ce WEEKDAY() foloseşte lunea drept prima zi a săptămânii, la poziţia 0 şi duminică drept ultima zi a săptămânii, la poziţia 6, rezultatul 0 este corect: luni.

Funcţiile DAYOFMONTH() şi DAYOFYEAR() (ziua din lună şi, respectiv, ziua din an) sunt mai clare, având un singur rezultat fiecare şi un interval care începe cu 1 şi se termină cu 31 pentru DAYOFMONTH() şi cu 366 pentru DAYOFYEAR(). Urmează câteva exemple:

```
mysql> select dayofmonth('2003-08-25');
+--------------------------+
| DAYOFMONTH('2003-08-25') |
+--------------------------+
|                       25 |
+--------------------------+
1 row in set (0.00 sec)

mysql> select dayofyear('2003-08-25');
+-------------------------+
| DAYOFYEAR('2003-08-25') |
+-------------------------+
|                     237 |
+-------------------------+
1 row in set (0.00 sec)
```

Poate părea ciudat că există o funcţie care returnează ziua din lună pentru o anumită dată, deoarece ziua este chiar acolo în şirul care reprezintă data calendaristică. Gândiţi-vă, însă, la situaţia în care folosiţi aceste funcţii în cadrul unor clauze WHERE pentru a face comparaţii între înregistrări. Dacă aveţi un tabel care conţine comenzi care au un câmp ce stochează data la care a fost primită comanda, puteţi să număraţi foarte uşor comenzile primite într-o anumită zi a săptămânii, sau să vedeţi câte comenzi au fost primite în prima jumătate a lunii faţă de câte au fost primite în cea de-a doua jumătate a lunii.

Cele două interogări de mai jos arată câte comenzi au fost primite în timpul primelor trei zile ale săptămânii (de-a lungul tuturor lunilor) şi, apoi, cele din restul zilelor săptămânii:

```
mysql> select count(id) from orders where dayofweek(date_ordered) < 4;
+-----------+
| COUNT(id) |
+-----------+
|         3 |
+-----------+
1 row in set (0.00 sec)

mysql> select count(id) from orders where dayofweek(date_ordered) > 3;
+-----------+
| COUNT(id) |
+-----------+
|         5 |
+-----------+
1 row in set (0.00 sec)
```

Folosind DAYOFMONTH(), următoarele exemple arată numărul de comenzi primite în prima jumătate a oricărei luni faţă de cele primite în cea de-a doua jumătate.

```
mysql> select count(id) from orders where dayofmonth(date_ordered) < 16;
+-----------+
| COUNT(id) |
+-----------+
|         6 |
+-----------+
1 row in set (0.00 sec)

mysql> select count(id) from orders where dayofmonth(date_ordered) > 15;
+-----------+
| COUNT(id) |
+-----------+
|         2 |
+-----------+
1 row in set (0.00 sec)
```

Puteţi folosi funcţia DAYNAME() pentru aduce rezultatele mai aproape de viaţa reală, această funcţie returnând numele zilei din săptămână (în engleză) pentru o anumită dată calendaristică.

```
mysql> select dayname(date_ordered) from orders;
+-----------------------+
| DAYNAME(date_ordered) |
+-----------------------+
| Thursday              |
| Monday                |
| Thursday              |
| Thursday              |
| Wednesday             |
| Thursday              |
| Sunday                |
| Sunday                |
+-----------------------+
8 rows in set (0.00 sec)
```

Aceste funcţii nu trebuie folosite neapărat în cadrul clauzelor WHERE — le puteţi folosi şi în clauze ORDER BY:

```
mysql> select dayname(date_ordered) from orders
    -> order by dayofweek(date_ordered);
+-----------------------+
| DAYNAME(date_ordered) |
+-----------------------+
| Sunday                |
| Sunday                |
| Monday                |
| Wednesday             |
| Thursday              |
| Thursday              |
| Thursday              |
| Thursday              |
+-----------------------+
8 rows in set (0.00 sec)
```

Operaţii cu luni şi ani

Zilele săptămânii reprezintă numai o parte din calendar, iar MySQL are şi funcţii concepute special pentru operaţii cu luni şi ani. La fel ca funcţiile DAYOFWEEK() şi DAYNAME(), funcţiile MONTH() şi MONTHNAME() returnează numărul lunii din an şi, respectiv, numele lunii pentru o anumită dată calendaristică. De exemplu:

```
mysql> select month('2003-08-25'), monthname('2003-08-25');
+--------------------+------------------------+
| month('2003-08-25') | monthname('2003-08-25') |
+--------------------+------------------------+
|                  8 | August                 |
+--------------------+------------------------+
1 row in set (0.00 sec)
```

Folosind MONTHNAME() asupra comenzilor din tabelă, interogarea din exemplul de mai jos afişează nişte rezultate corecte, dar care conţin foarte multe date redundante:

```
mysql> select monthname(date_ordered) from orders;
+-------------------------+
| MONTHNAME(date_ordered) |
+-------------------------+
| November                |
| November                |
| November                |
| November                |
| November                |
| November                |
| November                |
| October                 |
+-------------------------+
8 rows in set (0.00 sec)
```

Puteţi folosi clauza DISTINCT pentru a obţine rezultate care nu se repetă:

```
mysql> select distinct monthname(date_ordered) from orders;
+-------------------------+
| MONTHNAME(date_ordered) |
+-------------------------+
| November                |
| October                 |
+-------------------------+
2 rows in set (0.00 sec)
```

În ceea ce priveşte operaţiunile cu ani, funcţia YEAR() returnează anul corespunzător unei anumite date calendaristice:

```
mysql> select distinct year(date_ordered) from orders;
+--------------------+
| YEAR(date_ordered) |
+--------------------+
|               2003 |
+--------------------+
1 row in set (0.00 sec)
```

Operații cu săptămâni

Săptămânile pot pune probleme deosebite — un an poate avea 53 de săptămâni dacă prima zi a săptămânii este considerată duminică, iar ultima săptămână din decembrie nu s-a încheiat. De exemplu, 30 decembrie 2001 a picat într-o duminică:

```
mysql> select dayname('2001-12-30');
+-----------------------+
| DAYNAME('2001-12-30') |
+-----------------------+
| Sunday                |
+-----------------------+
1 row in set (0.00 sec)
```

Din această cauză, ziua de 30 decembrie 2001 face parte din cea de-a 53-a săptămână a anului:

```
mysql> select week('2001-12-30');
+--------------------+
| WEEK('2001-12-30') |
+--------------------+
|                 53 |
+--------------------+
1 row in set (0.00 sec)
```

Cea de-a 53-a săptămână conține zilele de 30 și 31 și, deci, este o săptămână de două zile; prima săptămână din 2002 începe cu 1 ianuarie.

Dacă vreți ca săptămânile să înceapă lunea, dar vreți, totuși, să aflați săptămâna dintr-un an corespunzătoare unei date calendaristice, un al doilea argument opțional vă permite să schimbați ziua de început a săptămânii. Valoarea 1 indică o săptămână care începe lunea. În exemplele de mai jos, folosirea zilei de luni ca început de săptămână face ca data de 30 decembrie să fie în cea de-a 52-a săptămână a anului 2001, dar data de 31 decembrie este, încă, în cea de-a 53-a săptămână a acestui an.

```
mysql> select week('2001-12-30',1);
+----------------------+
| WEEK('2001-12-30',1) |
+----------------------+
|                   52 |
+----------------------+
1 row in set (0.00 sec)

mysql> select week('2001-12-31',1);
+----------------------+
| WEEK('2001-12-31',1) |
+----------------------+
|                   53 |
+----------------------+
1 row in set (0.00 sec)
```

Operaţii cu ore, minute şi secunde

Dacă folosiţi o dată calendaristică ce conţine ora exactă, aşa cum ar fi cele de tipul `datetime` sau `timestamp`, sau chiar şi un câmp de tipul `time`, există funcţii cu care puteţi afla orele, minutele şi secundele din şirul de caractere respectiv. Deloc surprinzător, aceste funcţii se numesc `HOUR()`, `MINUTE()` şi `SECOND()`. Funcţia `HOUR()` returnează ora, care poate fi cuprinsă între 0 şi 23. Intervalul valorilor returnate de `MINUTE()` şi `SECOND()` este de la 0 la 59.

Iată câteva exemple:

```
mysql> select hour('2003-08-25 07:27:49') as hour,minute('2003-08-25 07:27:49')
    -> as minute,second('2003-08-25 07:27:49') as second;
+------+--------+--------+
| hour | minute | second |
+------+--------+--------+
|    7 |     27 |     49 |
+------+--------+--------+
1 row in set (0.00 sec)
```

Rezultă cam multe rezultate dintr-un singur câmp de tip `datetime` — aţi putea pune orele şi minutele împreună, sau aţi putea folosi chiar funcţia `CONCAT_WS()` pentru a introduce două puncte (:) între rezultate obţinând, astfel, o reprezentare umană a orei respective:

```
mysql> select concat_ws(':',hour('2003-08-25 07:27:49'),
    -> minute('2003-08-25 07:27:49')) as sample_time;
+-------------+
| sample_time |
+-------------+
| 7:27        |
+-------------+
1 row in set (0.00 sec)
```

Dacă folosiţi nume de câmpuri în loc de şiruri de caractere, amintiţi-vă că trebuie să nu folosiţi ghilimele. Iată un exemplu care foloseşte un câmp cu numele `dateadded` din presupusul tabel `sometable`:

```
mysql> select concat_ws(':',hour(dateadded), minute(dateadded))
    -> as sample_time from sometable;
+-------------+
| sample_time |
+-------------+
| 13:11       |
| 13:11       |
| 13:11       |
| 13:11       |
| 14:16       |
| 10:12       |
| 10:12       |
| 10:12       |
| 10:12       |
+-------------+
9 rows in set (0.00 sec)
```

Am trişat, pentru că rezultatele nu reprezintă ore de-adevărat — sunt doar două numere lipite unul lângă celălalt. Dacă aţi folosi şmecheria cu concatenarea asupra unei ore cum este `Ø2:Ø2`, rezultatul ar fi `2:2`, după cum se poate vedea mai jos:

```
mysql> select concat_ws(':',hour('Ø2:Ø2'), minute('Ø2:Ø2')) as sample_time;
+-------------+
| sample_time |
+-------------+
| 2:2         |
+-------------+
1 row in set (Ø.ØØ sec)
```

Rezultatul nu este, evident, cel dorit. În secţiunea următoare, veţi învăţa să folosiţi funcţia `DATE_FORMAT()` pentru a formata corespunzător datele calendaristice şi orele.

Formatarea datelor calendaristice şi a orelor în MySQL

Funcţia `DATE_FORMAT()` formatează un câmp de tipul `date`, `datetime` sau `timestamp` sub forma unui şir folosind opţiuni care indică exact modul în care rezultatele trebuie afişate. Sintaxa funcţiei `DATE_FORMAT()` este:

```
DATE_FORMAT(data, formatul)
```

Există multe opţiuni de formatare, după cum se poate vedea în tabelul 15.2:

TABELUL 15.2. Opţiunile de formatare a şirurilor pentru funcţia `DATE_FORMAT()`.

Opţiune	Rezultat
%M	Numele lunii, în engleză (de la January la December).
%b	Abrevierea numelui lunii, în engleză (de la Jan la Dec).
%m	Luna, în format de două cifre (de la 01 la 12).
%c	Luna (de la 1 la 12).
%W	Numele zilei săptămânii, în engleză (de la Sunday la Saturday).
%a	Abrevierea numelui zilei săptămânii, în engleză (de la Sun la Sat).
%D	Ziua din lună, folosind numeralul ordinal englezesc, adică first, second, third şi aşa mai departe.
%d	Ziua din lună, format de două cifre (de la 00 la 31).
%e	Ziua din lună (de la 0 la 31).
%j	Ziua din an, format de trei cifre (de la 001 la 366).
%Y	Anul, patru cifre.
%y	Anul, două cifre.
%X	Anul, în format de patru cifre, pentru care săptămâna în care duminică este prima zi; se foloseşte împreună cu %V.

TABELUL 15.2. *(continuare)*

Opțiune	Rezultat
%x	Anul, în format de patru cifre, pentru săptămâna în care luni este prima zi; se folosește împreună cu %v.
%w	Ziua din săptămână (0 = duminică ... 6 = sâmbătă).
%U	Săptămâna (de la 0 la 53) considerând că duminică este prima zi.
%u	Săptămâna (de la 0 la 53) considerând că luni este prima zi.
%V	Săptămâna (de la 1 la 53) considerând că duminică este prima zi; se folosește împreună cu %X.
%v	Săptămâna (de la 1 la 53) considerând că luni este prima zi; se folosește împreună cu %x.
%H	Ora, format de două cifre (de la 00 la 23).
%k	Ora (de la 0 la 23).
%h	Ora, format de două cifre (de la 01 la 12).
%l	Ora (de la 1 la 12).
%i	Minutele, format de două cifre (de la 00 la 59).
%S	Secundele, format de două cifre (de la 00 la 59).
%s	Secundele, format de două cifre (de la 00 la 59).
%r	Ora completă, în format de 12 ore (hh:mm:ss [AP]M).
%T	Ora completă, în format de 24 de ore (hh:mm:ss).
%p	AM sau PM.

Apropo

Orice alte caractere incluse în șirul de opțiuni pasat funcției `DATE_FORMAT()` vor apărea așa cum sunt.

Pentru a afișa rezultatul `02:02` pe care l-am falsificat în secțiunea anterioară, vom folosi opțiunile `%h` și `%i` pentru a returna ora și minutele corespunzătoare respectivei date, cu un semn „:" între cele două opțiuni. De exemplu:

```
mysql> select date_format('2003-08-25 02:02:00', '%h:%i') as sample_time;
+-------------+
| sample_time |
+-------------+
| 02:02       |
+-------------+
1 row in set (0.00 sec)
```

În cele ce urmează, sunt prezentate câteva exemple de folosire a funcției `DATE_FORMAT()`, dar trebuie menționat că nu puteți înțelege această funcție decât exersând singur folosirea ei.

```
mysql> select date_format('2003-08-25', '%W, %M %D, %Y') as sample_time;
+------------------------------+
| sample_time                  |
+------------------------------+
| Monday, August 25th, 2003    |
+------------------------------+
1 row in set (0.00 sec)

mysql> select date_format(now(),'%W the %D of %M, %Y around %l o\'clock %p')
    -> as sample_time;
+-------------------------------------------------------+
| sample_time                                           |
+-------------------------------------------------------+
| Monday the 25th of August, 2003 around 8 o'clock AM   |
+-------------------------------------------------------+
1 row in set (0.00 sec)
```

Dacă lucraţi cu câmpuri care conţin exclusiv ore, funcţia `TIME_FORMAT()` operează exact ca funcţia `DATE_FORMAT()`, dar nu sunt permise decât opţiunile privitoare la ore, minute şi secunde:

```
mysql> select time_format('02:02:00', '%h:%i') as sample_time;
+-------------+
| sample_time |
+-------------+
| 02:02       |
+-------------+
1 row in set (0.00 sec)
```

Calculele aritmetice cu date calendaristice în MySQL

MySQL are câteva funcţii care vă ajută să efectuaţi calcule aritmetice cu date calendaristice şi aceasta este unul dintre domeniile în care s-ar putea să fie mai rapid să faceţi calculele în MySQL decât în scriptul PHP. Funcţiile `DATE_ADD()` şi `DATE_SUB()` returnează un rezultat pe baza unei date de pornire şi a unui interval. Sintaxa ambelor funcţii este:

```
DATE_ADD(data, INTERVAL valoare tip)
DATE_SUB(data, INTERVAL valoare tip)
```

Tabelul 15.3 enumeră tipurile posibile şi formatul valorii care trebuie folosit pentru fiecare dintre ele.

TABELUL 15.3. Valorile şi tipurile implicate în calculele aritmetice cu date calendaristice.

Valoare	Tip
Numărul de secunde	SECOND
Numărul de minute	MINUTE
Numărul de ore	HOUR

TABELUL 15.3. (*continuare*)

Valoare	Tip
Numărul de zile	DAY
Numărul de luni	MONTH
Numărul de ani	YEAR
„minute:secunde"	MINUTE_SECOND
„ore:minute"	HOUR_MINUTE
„zile ore"	DAY_HOUR
„ani-luni"	YEAR_MONTH
„ore:minute:secunde"	HOUR_SECOND
„zile ore:minute"	DAY_MINUTE
„zile ore:minute:secunde"	DAY_SECOND

De exemplu, pentru a afla data de peste 21 de zile, folosiţi următoarea interogare:

```
mysql> select date_add(now(), interval 21 day);
+----------------------------------+
| date_add(now(), interval 21 day) |
+----------------------------------+
| 2003-09-14 16:22:27              |
+----------------------------------+
1 row in set (0.00 sec)
```

Pentru a scădea 21 de zile, folosiţi:

```
mysql> select date_sub(now(), interval 21 day);
+----------------------------------+
| date_sub(now(), interval 21 day) |
+----------------------------------+
| 2003-08-03 16:22:38              |
+----------------------------------+
1 row in set (0.00 sec)
```

Folosiţi expresia aşa cum este indicată în tabelul 15.3, chiar dacă vi se pare natural să scrieţi DAYS (zile), în loc de DAY (zi). Dacă folosiţi DAYS, veţi provoca o eroare:

```
mysql> select date_add(now(), interval 21 days);
ERROR 1064: You have an error in your SQL syntax near 'days)' at line 1
```

Dacă folosiţi DATE_ADD() şi DATE_SUB() cu valoare de tip date în locul unei valori de tip datetime, rezultatul va fi afişat ca o valoarea de tip date cu excepţia cazului în care folosiţi expresii care se referă la ore, minute şi secunde. În acest caz, rezultatul va fi o valoare de tipul datetime.

De exemplu, rezultatul primei interogări de mai jos rămâne o valoare de tip date, pe când cea de-a doua are un rezultat de tip datetime:

```
mysql> select date_add("2001-12-31", interval 1 day);
+-----------------------------------------+
| DATE_ADD("2001-12-31", INTERVAL 1 DAY) |
+-----------------------------------------+
| 2002-01-01                              |
+-----------------------------------------+
1 row in set (0.00 sec)

mysql> select date_add("2001-12-31", interval 12 hour);
+-----------------------------------------+
| DATE_ADD("2001-12-31", INTERVAL 12 HOUR) |
+-----------------------------------------+
| 2001-12-31 12:00:00                     |
+-----------------------------------------+
1 row in set (0.00 sec)
```

Începând de la versiunea 3.23, MySQL poate efectua calcule cu date calendaristice şi folosind operatorii + şi – în locul funcţiilor `DATE_ADD()` şi `DATE_SUB()`:

```
mysql> select "2001-12-31" + interval 1 day;
+-------------------------------+
| "2001-12-31" + INTERVAL 1 DAY |
+-------------------------------+
| 2002-01-01                    |
+-------------------------------+
1 row in set (0.00 sec)

mysql> select "2001-12-31" - interval 14 hour;
+---------------------------------+
| "2001-12-31" - INTERVAL 14 HOUR |
+---------------------------------+
| 2001-12-30 10:00:00             |
+---------------------------------+
1 row in set (0.00 sec)
```

Funcţii speciale şi operaţiuni de conversie

Funcţia `NOW()` din MySQL returnează data şi ora curentă în format `datetime` şi este utilă pentru înregistrarea momentului autentificării unui utilizator sau pentru păstrarea duratei accesului, precum şi în multe alte scopuri. MySQL mai alte câteva funcţii care se comportă similar.

Funcţiile `CURDATE()` şi `CURRENT_DATE()` sunt sinonime, ambele returnând data curentă în formatul `AAAA-LL-ZZ` (`YYYY-MM-DD`, în limba engleză):

```
mysql> select curdate(), current_date();
+------------+---------------+
| curdate()  | current_date() |
+------------+---------------+
| 2003-08-25 | 2003-08-25    |
+------------+---------------+
1 row in set (0.01 sec)
```

În mod similar, funcţiile CURTIME() şi CURRENT_TIME() returnează ora curentă în formatul HH:MM:SS:

```
mysql> select curtime(), current_time();
+-----------+---------------+
| curtime() | current_time() |
+-----------+---------------+
| 09:14:26  | 09:14:26       |
+-----------+---------------+
1 row in set (0.00 sec)
```

Funcţiile NOW(), SYSDATE() şi CURRENT_TIMESTAMP() returnează valorile în format de datetime complet (AAAA-LL-ZZ HH:MM:SS):

```
mysql> select now(), sysdate(), current_timestamp();
+---------------------+---------------------+---------------------+
| now()               | sysdate()           | current_timestamp() |
+---------------------+---------------------+---------------------+
| 2003-08-25 09:14:50 | 2003-08-25 09:14:50 | 2003-08-25 09:14:50 |
+---------------------+---------------------+---------------------+
1 row in set (0.00 sec)
```

Funcţia UNIX_TIMESTAMP() au returnează data curentă — sau converteşte o dată care îi este pasată — în formatul de *timestamp* din Unix. Acest format este exprimat în secundele scurse de la începutul epocii Unix, adică miezul nopţii de 1 ianuarie 1970. De exemplu:

```
mysql> select unix_timestamp();
+------------------+
| UNIX_TIMESTAMP() |
+------------------+
|       1061767446 |
+------------------+
1 row in set (0.00 sec)

mysql> select unix_timestamp('1973-12-30');
+-----------------------------+
| UNIX_TIMESTAMP('1973-12-30') |
+-----------------------------+
|                  126086400 |
+-----------------------------+
1 row in set (0.00 sec)
```

Funcţia FROM_UNIXTIME() converteşte un *timestamp* Unix în formatul datetime atunci când este folosită fără alte opţiuni:

```
mysql> select from_unixtime('1061767446');
+----------------------------+
| from_unixtime('1061767446') |
+----------------------------+
| 2003-08-24 16:24:06        |
+----------------------------+
1 row in set (0.00 sec)
```

Puteţi folosi opţiunile de formatare de la funcţia DATE_FORMAT() pentru a afişa *timestamp*-ul într-un mod mai atrăgător:

```
mysql> select from_unixtime(unix_timestamp(), '%D %M %Y at %h:%i:%s');
+-----------------------------------------------------------+
| from_unixtime(unix_timestamp(), '%D %M %Y at %h:%i:%s') |
+-----------------------------------------------------------+
| 24th August 2003 at 04:24:54                              |
+-----------------------------------------------------------+
1 row in set (0.00 sec)
```

Dacă lucraţi cu un număr de secunde şi vreţi să convertiţi secundele într-un rezultat formatat, puteţi folosi funcţiile SEC_TO_TIME() şi TIME_TO_SEC() pentru a converti valorile într-un sens şi în celălalt.

De exemplu, 1 440 de secunde înseamnă 24 de minute şi viceversa:

```
mysql> select sec_to_time('1440'), time_to_sec('00:24:00');
+---------------------+-------------------------+
| SEC_TO_TIME('1440') | TIME_TO_SEC('00:24:00') |
+---------------------+-------------------------+
| 00:24:00            |                    1440 |
+---------------------+-------------------------+
1 row in set (0.01 sec)
```

Rezumat

În acest capitol, aţi învăţat elementele de bază ale limbajului SQL, începând de la crearea tabelelor şi terminând cu manipularea înregistrărilor. Comanda de creare a unei tabele necesită trei informaţii importante: numele tabelului, numele câmpurilor şi definiţiile câmpurilor. Definiţiile câmpurilor sunt importante pentru că o tabelă proiectată bine va contribui la viteza de utilizării bazei dumneavoastră de date. MySQL are trei categorii diferite de tipuri de date: numerice, calendaristice şi şiruri de caractere.

Comanda INSERT, folosită pentru adăugarea de înregistrări într-o tabelă, trebuie să specifice tabela şi coloanele pe care doriţi să le populaţi şi, apoi, să definească valorile. Atunci când amplasaţi valori cu ajutorul comenzii INSERT, şirurile trebuie să fie puse în ghilimele simple sau duble. Comanda SELECT din limbajul SQL este folosită pentru extragerea înregistrărilor dintr-o anumită tabelă. Caracterul * vă permite să selectaţi cu uşurinţă toate câmpurile tuturor înregistrărilor dintr-o tabelă. Dacă rezultatul este prea lung, clauza LIMIT reprezintă o metodă simplă de extragere a unor rezultate parţiale, permiţându-vă să specificaţi o poziţie de start şi numărul de înregistrări care trebuie returnate. Pentru a ordona rezultatele, puteţi folosi clauza ORDER BY selectând coloanele în funcţie de care să fie efectuată sortarea. Sortările se pot face în funcţie de numere întregi, date calendaristice sau şiruri, atât în ordine crescătoare cât şi în ordine descrescătoare. Ordinea implicită este cea crescătoare. Dacă nu specificaţi nici o ordine, rezultatele sunt afişate în ordinea în care apar în tabelă.

Puteţi să alegeţi ce înregistrări să fie returnate folosind clauze WHERE care verifică îndeplinirea unor condiţii. În cadrul clauzelor WHERE pot fi folosite comparaţii sau operatori logici, iar, uneori, ambele pot face parte din expresii complexe. Selectarea înregistrărilor din mai multe tabele în cadrul unei singure instrucţiuni reprezintă cea mai avansată metodă de lucru deoarece acest tip de instrucţiuni — denumite JOIN-uri — necesită prevedere şi planificare pentru a avea rezultate corecte. Tipurile cele mai uzuale de JOIN sunt INNER JOIN, LEFT JOIN şi RIGHT JOIN, deşi MySQL mai poate lucra cu multe alte feluri de JOIN.

Comenzile UPDATE şi REPLACE sunt folosite pentru modificarea datelor care există deja în tablele dumneavoastră MySQL. Comanda UPDATE este utilă pentru schimbarea valorilor din anumite coloane şi pentru schimbarea valorilor din mai multe înregistrări pe baza unor condiţii specifice. Comanda REPLACE este o variantă a comenzii INSERT care şterge şi apoi re-inserează o înregistrare pe baza valorii cheii primare. Fiţi foarte atenţi când folosiţi UPDATE pentru a modifica valorile dintr-o coloană deoarece, în lipsa unei condiţii, coloana respectivă va fi actualizată în cadrul tuturor înregistrărilor dintr-o tabelă.

Comanda DELETE este simplă — ea elimină toate înregistrările din tabele. Din această cauză, ea este şi foarte periculoasă, aşa că asiguraţi-vă că nu le daţi privilegii de folosire a comenzii DELETE decât utilizatorilor care se pot descurca cu o astfel de responsabilitate. Puteţi specifica condiţii atunci când folosiţi DELETE astfel încât înregistrările să fie eliminate numai dacă o este satisfăcută o anumită condiţie dintr-o clauză WHERE. De asemenea, puteţi un anumit număr de înregistrări din tabela dumneavoastră folosind clauza LIMIT. Dacă aveţi o tabelă extrem de mare, ştergerea pe porţiuni consumă mai puţine resurse decât ştergerea tuturor înregistrărilor dintr-o astfel de tabelă uriaşă.

Vi s-au prezentat funcţiile MySQL care operează cu şiruri de caractere, date calendaristice şi ore. Dacă vreţi să concatenaţi şiruri de caractere în MySQL sau dacă vreţi să număraţi caracterele dintr-un şir, puteţi folosi funcţii precum CONCAT(), CONCAT_WS() şi, respectiv, LENGTH(). Pentru a adăuga sau elimina caractere de completare dintr-un şir, puteţi folosi funcţiile RPAD(), LPAD(), TRIM(), LTRIM() şi RTRIM() pentru a aduce şirul dumneavoastră exact în forma care vă trebuie. Puteţi, de asemenea, să găsiţi poziţia unui şir în interiorul altuia, sau să preluaţi o parte dintr-un şir dat folosind funcţiile LOCATE(), SUBSTRING(), LEFT() şi RIGHT(). Funcţii precum LCASE(), UCASE(), REPEAT() şi REPLACE() returnează şi ele variaţiuni ale şirurilor originale. Funcţiile de lucru cu date calendaristice şi ore care sunt implementate în MySQL pot, fără îndoială, să preia o parte din sarcinile aplicaţiei dumneavoastră formatând intern datele şi orele şi efectuând calcule aritmetice cu date calendaristice. Opţiunile de formatare folosite împreună cu funcţia DATE_FORMAT() vă oferă o metodă simplă de afişare personalizată a oricărui tip de câmp ce conţine date calendaristice. Funcţiile DATE_ADD() şi DATE_SUB(), împreună cu numeroasele tipuri de intervale pe care le pot folosi, vă ajută să găsiţi date calendaristice din trecut şi din viitor. În plus, funcţii precum DAY(), WEEK(), MONTH() şi YEAR() sunt utile pentru extragerea unor părţi ale datelor calendaristice în vederea folosirii acestora în clauze WHERE sau ORDER BY.

Întrebări şi răspunsuri

Î. *Ce caractere pot folosi pentru a denumi tabelele şi câmpurile şi care este limita de caractere?*

R. Lungimea maximă a numelui unei baze de date, a unui câmp şi a unei tabele este de 64 de caractere. Orice caracter ce poate fi folosit în numele unui director sau al unui fişier poate fi folosit în numele bazelor de date sau tabelelor — cu excepţia caracterelor / şi .. Aceste limitări există deoarece MySQL creează directoare şi fişiere ce corespund bazelor de date şi tabelelor în sistemul dumneavoastră de fişiere. În afară de lungime, nu există alte limitări legate de numele câmpurilor.

Î. *Pot folosi mai multe funcţii într-o singură instrucţiune, cum ar fi, de exemplu să transform un şir concatenat în litere mari?*

R. Desigur — fiţi atenţi, totuşi, la deschiderea şi închiderea parantezelor. Exemplul de mai jos vă arată cum să transformaţi în litere mari un şir de caractere rezultat din concatenarea numelor şi prenumelor dintr-o tabelă.

```
mysql> SELECT UCASE(CONCAT_WS(' ', firstname, lastname)) FROM
table_name;
+-----------------------------------------+
| UCASE(CONCAT_WS(' ', firstname, lastname)) |
+-----------------------------------------+
| JOHN SMITH                              |
| JANE SMITH                              |
| JIMBO JONES                             |
| ANDY SMITH                              |
| CHRIS JONES                             |
| ANNA BELL                               |
| JIMMY CARR                              |
| ALBERT SMITH                            |
| JOHN DOE                                |
+-----------------------------------------+
9 rows in set (0.00 sec)
```

Dacă vreţi să transformaţi în litere mari numai numele de familie, puteţi folosi:

```
mysql> SELECT CONCAT_WS(' ', firstname, UCASE(lastname)) FROM
master_name;
+-----------------------------------------+
| CONCAT_WS(' ', firstname, UCASE(lastname)) |
+-----------------------------------------+
| John SMITH                              |
| Jane SMITH                              |
| Jimbo JONES                             |
| Andy SMITH                              |
| Chris JONES                             |
| Anna BELL                               |
| Jimmy CARR                              |
| Albert SMITH                            |
| John DOE                                |
+-----------------------------------------+
9 rows in set (0.00 sec)
```

Atelier

Atelierul este conceput să anticipeze posibile întrebări, să vă ajute să recapitulaţi cele învăţate şi să începeţi să vă puneţi cunoştinţele în practică.

Test

1. Căror tipuri de date le-ar putea aparţine numărul întreg 56678685?
2. Cum aţi defini un câmp care nu va conţine decât următoarele şiruri: `mar`, `para`, `banana`, `cireasa`?
3. Care ar fi clauza `LIMIT` pentru selectarea primelor 25 de înregistrări dintr-o tabelă? Dar pentru următoarele 25?
4. Cum aţi formula o comparaţie între şiruri de caractere, folosind `LIKE`, pentru ca ea să găsească atât prenumele `"Ion"`, cât şi prenumele `"Ioana"`?
5. Cum v-aţi referi în mod explicit la câmpul numit `id` dintr-o tabelă numită `table1`?
6. Scrieţi o instrucţiune SQL care să facă o joncţiune între două tabele, `comenzi` şi `obiecte_comandate`, pe baza câmpului `id_comanda` care este cheie primară în ambele. Din tabela comenzi, selectaţi următoarele câmpuri: `nume_comanda` şi `data_comenzii`. Din tabela `obiecte_comandate`, selectaţi câmpul `descriere_obiect`.
7. Scrieţi o interogare SQL care să găsească poziţia la care începe şirul `"strugure"` în interiorul şirului `"marparabananastrugure"`.
8. Scrieţi o interogare care să selecteze ultimele cinci caractere din şirul `"marparabananastrugure"`.

Soluţii

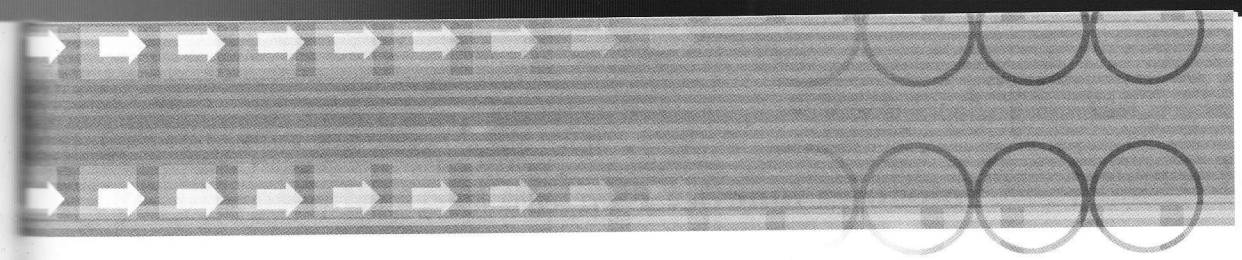
1. `MEDIUMINT`, `INT` sau `BIGNINT`.
2. `ENUM('mar', 'para ', 'banana ', 'cireasa ')` sau `SET('mar', 'para ', 'banana ', 'cireasa ')`
3. `LIMIT 0,25` şi `LIMIT 25,25`
4. `LIKE 'Io%'`
5. Folosiţi expresia `table1.id` în loc de `id` în interogarea dumneavoastră.
6.
```
SELECT comenzi.nume_comanda, comenzi.data_comenzii,
obiecte_comandate.descriere_obiect FROM comenzi LEFT JOIN obiecte_
comandate ON comenzi.id_comanda =
obiecte_comandate.id_comanda;
```

7. `SELECT LOCATE("strugure", "marparabananastrugure");`

8. `SELECT RIGHT("marparabananastrugure",5);`

Activitate

Petreceţi ceva vreme creând tabele de probă şi exersaţi folosirea comenzilor `INSERT` şi `SELECT`.



CAPITOLUL 16
Interacțiunea între scripturile PHP şi MySQL

Acum că aţi învăţat atât elementele de bază ale limbajului PHP, cât şi cele ale folosirii pachetului MySQL, sunteţi gata să exersaţi interacţiunea dintre cele două. Gândiţi-vă la PHP ca la o interfaţă pentru MySQL — comenzile pe care le veţi trimite către serverul MySQL în acest capitol sunt cele pe care le-aţi învăţat în capitolul precedent, doar că acum le veţi trimite prin intermediul scripturilor PHP.

În acest capitol, veţi învăţa cum să:

- vă conectaţi la serverul MySQL folosind un script PHP;
- inseraţi şi să selectaţi date prin intermediul scripturilor PHP.

Conectarea la MySQL folosind PHP

Pentru a reuşi să folosiţi funcţiile PHP pentru a comunica cu MySQL, trebuie ca serverul MySQL să ruleze pe un calculator la care serverul de Web să se poată conecta (nu neapărat pe aceeaşi maşină ca serverul Web). Trebuie, de asemenea, să fi creat un utilizator (cu o parolă) şi trebuie să ştiţi numele bazei de date la care vreţi să vă conectaţi. Dacă aţi urmat instrucţiunile din capitolul 1, intitulat *Instalarea şi configurarea serverului MySQL*, şi din capitolul 3, *Instalarea şi configurarea pachetului PHP*, aceste lucruri sunt deja rezolvate.

În toate scripturile ilustrative din acest capitol, baza de date folosită se numeşte `testDB`, numele de utilizator este `joeuser`, iar parola `somepass`. Substituiţi aceste valori cu informaţiile dumneavoastră atunci când folosiţi aceste scripturi.

Apropo

Puteți găsi secțiunea din Manualul PHP care prezintă toate funcțiile de lucru cu MySQL la adresa `http://www.php.net/mysql`. Folosiți această secțiune!

Folosirea funcției `mysql_connect()`

Funcţia `mysql_connect()` este prima care trebuie apelată atunci când folosiţi un script PHP pentru conectarea la MySQL — dacă nu deschideţi o conexiune către serverul MySQL, nu o să ajungeţi prea departe! Sintaxa de bază pentru stabilirea conexiunii este:

```
mysql_connect("localhost", "utilizator", "parola");
```

Folosind valorile efective, funcţia de conectare arată astfel:

```
mysql_connect("localhost", "joeuser", "somepass");
```

În cazul în care conexiunea este stabilită cu succes, funcţia returnează un indice al conexiunii, iar, în caz contrar, returnează valoarea false. Listingul 16.1 este un exemplu funcţional de script care stabileşte o conexiune. El atribuie valoarea indicelui conexiunii unei variabile numită $conn şi, apoi, afişează valoarea variabilei $conn drept dovadă a existenţei conexiunii.

LISTINGUL 16.1. Un script simplu care stabileşte o conexiune.

```
1: <?php
2: $conn = mysql_connect("localhost", "joeuser", "somepass");
3: echo $conn;
4: ?>
```

Salvaţi acest script sub numele `mysqlconnect.php` şi amplasaţi-l în zona de documente a serverului dumneavoastră Web. Accesaţi scriptul cu ajutorul browserului şi veţi vedea ceva asemănător:

```
Resource id #1
```

Conectarea la MySQL prin intermediul funcţiei `mysql_connect()` este o operaţiune destul de simplă. Conexiunea este încheiată când scriptul îşi termină execuţia, dar dacă vreţi să încheiaţi conexiunea în mod explicit, puteţi să adăugaţi un apel către funcţia `mysql_close()` la sfârşitul scriptului, ca în listingul 16.2.

LISTINGUL 16.2. Scriptul simplu de stabilire a conexiunii modificat.

```
1: <?php
2: $conn = mysql_connect("localhost", "joeuser", "somepass");
3: echo $conn;
4: mysql_close($conn);
5: ?>
```

Asta e tot. Secţiunea următoare se ocupă de funcţiile de executare a interogărilor, care sunt mult mai interesante decât simpla deschidere a conexiunii de dragul de a o deschide.

Executarea interogărilor

Jumătate din rezolvarea problemei executării interogărilor MySQL din PHP constă în capacitatea de scriere a codului SQL efectiv. Funcţia PHP `mysql_query()` este folosită pentru a trimite interogarea dumneavoastră SQL către serverul MySQL. Dacă acest lucru reuşeşte, este returnat un indice al rezultatului. Dacă funcţia eşuează, ea returnează valoarea `false`.

Atunci când folosiţi funcţia `mysql_query()`, veţi observa că lipseşte un element: selectarea bazei de date care trebuie folosită. Atunci când vă conectaţi la MySQL cu ajutorul interfeţei în linie de comandă, baza de date este specificată în între argumentele liniei pe care o scrieţi pentru a vă conecta, sau poate fi schimbată manual după ce vă autentificaţi. În PHP, acest lucru se face prin intermediul unei alte funcţii denumite `mysql_select_db()` care are următoarea sintaxă:

```
mysql_select_db(nume_baza_de_date, indice_conexiune);
```

Pentru a vă conecta la o bază de date cu numele `testDB`, mai întâi folosiţi funcţia `mysql_connect()` şi, apoi, funcţia `mysql_select_db()`, după cum se poate vedea în listingul 16.3.

LISTINGUL 16.3. Conectarea şi selectarea bazei de date.

```
1: <?php
2: $conn = mysql_connect("localhost", "joeuser", "somepass");
3: mysql_select_db("testDB",$conn);
4: ?>
```

Acum aveţi două informaţii importante: indicele conexiunii (`$conn`) şi siguranţa faptului că PHP va folosi baza de date `testDB` pe întreaga durată a execuţiei scriptului respectiv. Indicele conexiunii este folosit în sintaxa funcţiei `mysql_query()`:

```
mysql_query(interogare, indice_conexiune);
```

În scriptul dumneavoastră, mai întâi stabiliţi conexiunea, apoi executaţi o interogare. Scriptul din listingul 16.4 creează o tabelă simplă denumită `testTable`.

LISTINGUL 16.4. Un script care creează o tabelă.

```
 1: <?php
 2: // deschide conexiunea
 3: $conn = mysql_connect("localhost", "joeuser", "somepass");
 4: // alege baza de date care va fi folosită
 5: mysql_select_db("testDB",$conn);
 6: // compune instrucţiunea SQL
 7: $sql = "CREATE TABLE testTable (id int not null primary key auto_increment,
 8: testField varchar (75))";
 9: // execută instrucţiunea SQL
10: $result = mysql_query($sql, $conn);
11: // afişează identificatorul rezultatului
12: echo $result;
13: ?>
```

Apropo

> Când trimiteți interogări prin intermediul funcției `mysql_query()`, caracterul punct și virgulă de la sfârșitul instrucțiunilor SQL nu mai este necesar. Singurul caracter punct și virgulă ce trebuie să existe în linia de cod respectivă este cea de la sfârșitul comenzii PHP.

Din cauză că funcția `mysql_query()` returnează numai valorile `true` sau `false`, rezultatul afișat de script este tare neinteresant:

```
1
```

Valoarea `1` indică, de fapt, valoarea `true`, adică faptul că interogarea a fost executată cu succes. Valoarea `0` ar fi indicat eșecul. Intrați în MySQL cu interfața în linie de comandă pentru a verifica crearea tabelei `testTable`:

```
mysql> describe testTable;
+-----------+-------------+------+-----+---------+----------------+
| Field     | Type        | Null | Key | Default | Extra          |
+-----------+-------------+------+-----+---------+----------------+
| id        | int(11)     |      | PRI | NULL    | auto_increment |
| testField | varchar(75) | YES  |     | NULL    |                |
+-----------+-------------+------+-----+---------+----------------+
2 rows in set (0.00 sec)
```

Felicitări — ați reușit să creați o tabelă în baza dumneavoastră de date MySQL prin intermediul unui script PHP!

Preluarea mesajelor de eroare

Petreceți ceva timp familiarizându-vă cu funcția `mysql_error()` — ea vă va fi cea mai bună prietenă. Atunci când este folosită împreună cu funcția PHP `die()`, care pur și simplu oprește execuția scriptului în momentul în care este apelată, funcția `mysql_error()` va returna mesajul de eroare care este foarte util atunci când faceți vreo greșeală.

De exemplu, din moment ce ați creat deja o tabelă numită `testTable`, nu veți mai putea executa scriptul acela din nou fără a provoca o eroare. Haideți să încercăm să executăm din nou scriptul, dar mai întâi să îl modificăm pentru a utiliza funcția `mysql_error()` (vezi listingul 16.5).

LISTINGUL 16.5. Scriptul de creare a tabelei, cu mesaje de eroare.

```
1: <?php
2: // deschide conexiunea
3: $conn = mysql_connect("localhost", "joeuser", "somepass");
4: // alege baza de date care va fi folosită
5: mysql_select_db("testDB",$conn);
6: // compune instrucţiunea SQL
7: $sql = "CREATE TABLE testTable (id int not null primary key auto_increment,
8: testField varchar (75))";
9: // execută instrucţiunea SQL
10: $result = mysql_query($sql, $conn) or die(mysql_error());
```

LISTINGUL 16.5. (*continuare*)

```
11: // afișează identificatorul rezultatului
12: echo $result;
13: ?>
```

Atunci când executați scriptul, ar trebui să vedeți în browser ceva similar cu:

```
Table 'testTable' already exists
```

Cât de interesant! Treceți acum la secțiunea următoare pentru a începe să inserați date în tabela dumneavoastră și, foarte curând, veți prelua date din baza de date și le veți formata în PHP.

Lucrul cu datele MySQL

Inserarea, actualizarea, ștergerea și extragerea datelor, toate se bazează pe funcția `mysql_query()`, care execută interogări SQL. Pentru `INSERT`, `UPDATE` și `DELETE`, nu este nevoie să scrieți cod după ce ați trimis interogarea pentru că nu trebuie să afișați nici un rezultat (decât dacă doriți să o faceți). Pentru `SELECT`, aveți câteva opțiuni de afișare a datelor extrase de interogarea dumneavoastră. Haideți să începem cu lucruri simple și să inserăm niște date, astfel încât să aveți ce să extrageți mai târziu.

Inserarea datelor prin intermediul PHP

Cea mai ușoară metodă de inserare a datelor este codificarea statică a instrucțiunii `INSERT`, așa cum se vede în listingul 16.6.

LISTINGUL 16.6. Un script care inserează o înregistrare.

```
 1: <?php
 2: // deschide conexiunea
 3: $conn = mysql_connect("localhost", "joeuser", "somepass");
 4: // alege baza de date care va fi folosită
 5: mysql_select_db("testDB",$conn);
 6: // compune instrucțiunea SQL
 7: $sql = "INSERT INTO testTable values ('', 'some value')";
 9: // execută instrucțiunea SQL
 9: $result = mysql_query($sql, $conn) or die(mysql_error());
10: // afișează identificatorul rezultatului
11: echo $result;
12: ?>
```

Ați putea să vă întrebați de ce trebuie să afișați identificatorul rezultatului din moment ce nu faceți decât să introduceți date. Păi, nu trebuie; am făcut-o doar așa. Puteți să curățați puțin acest script înlocuind linia care execută interogarea

astfel încât să nu facă altceva decât să se execute şi să afişeze o informare în cazul în care inserarea se face cu succes, aşa ca în listingul 16.7.

LISTINGUL 16.7. Scriptul de inserare modificat.

```
 1: <?php
 2: // deschide conexiunea
 3: $conn = mysql_connect("localhost", "joeuser", "somepass");
 4: // alege baza de date care va fi folosită
 5: mysql_select_db("testDB",$conn);
 6: // compune instrucţiunea SQL
 7: $sql = "INSERT INTO testTable values ('', 'some value')";
 8: // execută instrucţiunea SQL
 9: if (mysql_query($sql, $conn)) {
10:     echo "record added!";
11: } else {
12:     echo "something went wrong";
13: }
14: ?>
```

Rularea acestui script va avea drept rezultat adăugarea unui rând în tabela `testTable`. Pentru a introduce mai multe înregistrări, puteţi fie să scrieţi o listă lungă de instrucţiuni SQL statice şi să folosiţi funcţia `mysql_query()` de multe ori ca să executaţi aceste instrucţiuni, fie puteţi crea o interfaţă bazată pe un formular pentru scriptul de adăugare de înregistrări.

Pentru a crea un formular pentru acest script, nu veţi avea nevoie decât de un singur câmp, deoarece câmpul `id` poate fi incrementat automat. Acţiunea formularului este numele scriptului de adăugare a înregistrărilor; haideţi să îi spunem `insert.php`. Formularul HTML va arăta asemănător cu cel din listingul 16.8.

LISTINGUL 16.8. Un formular de inserare.

```
 1: <HTML>
 2: <HEAD>
 3: <TITLE>Insert Form</TITLE>
 4: </HEAD>
 5: <BODY>
 6: <FORM ACTION="insert.php" METHOD=POST>
 7: <P>Text to add:<br>
 8: <input type=text name="testField" size=30>
 9: <p><input type=submit name="submit" value="Insert Record"></p>
10: </FORM>
11: </BODY>
12: </HTML>
```

Salvaţi acest fişier sub numele `insert_form.html` şi puneţi-l în zona de documente a serverului dumneavoastră Web. Apoi, creaţi scriptul `insert.php` pe baza listingului 16.9. Valoarea introdusă în formular va înlocui valorile codificate

static în interogarea SQL cu o variabilă denumită `$_POST[testField]`.

LISTINGUL 16.9. Un script de inserare care foloseşte un formular.

```
 1: <?php
 2: // deschide conexiunea
 3: $conn = mysql_connect("localhost", "joeuser", "somepass");
 4: // alege baza de date care va fi folosită
 5: mysql_select_db("testDB",$conn);
 6: // compune instrucţiunea SQL
 7: $sql = "INSERT INTO testTable values ('', '$_POST[testField]')";
 8: // execută instrucţiunea SQL
 9: if (mysql_query($sql, $conn)) {
10:     echo "record added!";
11: } else {
12:     echo "something went wrong";
13: }
14: ?>
```

Salvaţi scriptul sub numele `insert.php` şi puneţi-l în zona de documente a serverului dumneavoastră Web. Accesaţi, în browserul dumneavoastră, formularul HTML pe care l-aţi creat. Ar trebui să arate ca în figura 16.1.

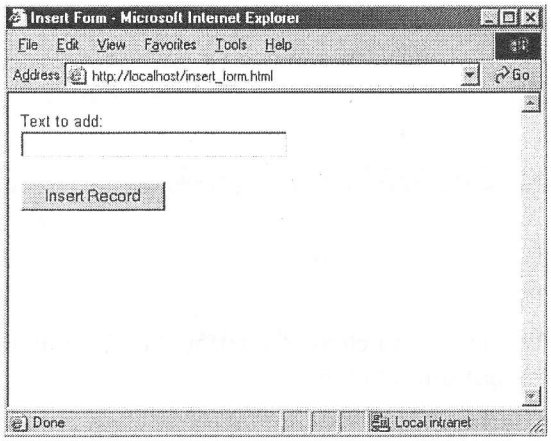


FIGURA 16.1. Formularul HTML pentru adăugarea unei înregistrări.

Introduceţi un şir de caractere în câmpul `Text to Add` (text de adăugat), aşa cum se vede în figura 16.2.

În final, daţi clic pe butonul `Insert Record` (inserează înregistrarea) ca să executaţi scriptul `insert.php` şi să inseraţi înregistrarea. Dacă acest lucru are loc cu succes, veţi vedea un rezultat asemănător cu cel din figura 16.3.

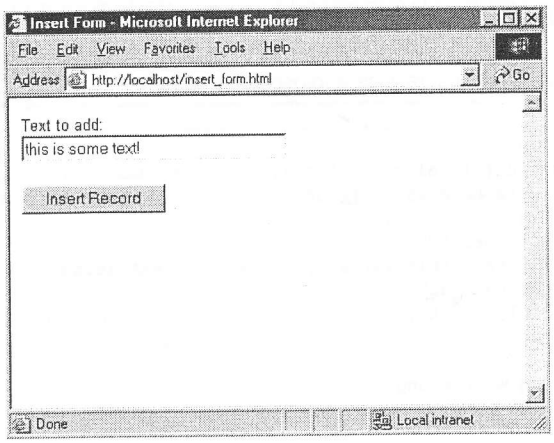


FIGURA 16.2. Textul introdus în câmpul formularului.

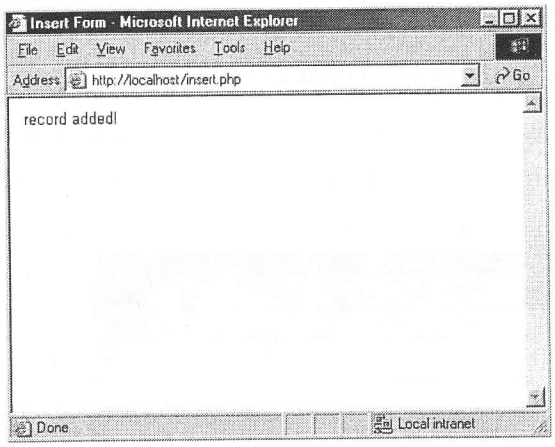


FIGURA 16.3. Înregistrarea a fost inserată cu succes.

Pentru a verifica ce aţi făcut, puteţi folosi interfaţa MySQL în linie de comandă pentru a vizualiza înregistrările din tabelă:

```
mysql> select * from testTable;
+----+--------------------+
| id | testField          |
+----+--------------------+
|  1 | some value         |
|  2 | this is some text! |
+----+--------------------+
2 rows in set (0.00 sec)
```

În continuare, veţi învăţa să extrageţi date şi să formataţi rezultatul în PHP.

Extragerea datelor în PHP

Pentru că aveți câteva înregistrări în tabela dumneavoastră `testTable`, puteți scrie un script PHP care să extragă date. Începeți cu ceva simplu și scrieți un script care trimite o interogare `SELECT` dar nu vă inundă cu date rezultate; haideți să preluăm doar numărul de înregistrări. Pentru a face asta, folosiți funcția `mysql_num_rows()`. Această funcție returnează un rezultat, așa că atunci când executați interogarea, puneți indicele rezultatului în variabila `$result` (vezi listingul 16.10).

LISTINGUL 16.10. Un script care extrage date.

```
<?php
// deschide conexiunea
$conn = mysql_connect("localhost", "joeuser", "somepass");
// alege baza de date care va fi folosită
mysql_select_db("testDB",$conn);
// compune instrucțiunea SQL
$sql = "SELECT * FROM testTable";
// execută instrucțiunea SQL
$result = mysql_query($sql, $conn) or die(mysql_error());
//preia numarul de inregistrari din tabelul de rezultate
$number_of_rows = mysql_num_rows($result);
echo "The number of rows is $number_of_rows";
?>
```

Salvați acest script sub numele `count.php` și puneți-l în zona de documente a serverului dumneavoastră Web. Ar trebui să vedeți un mesaj ca cel de mai jos:

```
The number of rows is 2
```

Numărul ar trebui să fie egal cu numărul de înregistrări pe care le-ați inserat în timpul testelor din secțiunea precedentă. Acum că știți că există câteva înregistrări în tabelă, puteți să treceți la lucruri serioase și să aduceți conținutul efectiv al acelor înregistrări. Puteți face asta în mai multe feluri, dar cea mai ușoară metodă este preluarea fiecărei înregistrări într-un vector.

Cea ce veți face este să folosiți o instrucțiune while pentru a parcurge secvențial fiecare înregistrare din setul de rezultate, atribuind valoarea fiecărui câmp unei variabile specifice și, apoi, veți afișa rezultatele pe ecran. Sintaxa funcției `mysql_fetch_array()` este:

```
$vectorNou = mysql_fetch_array($indice_rezultat)
```

Continuați folosind listingul 16.11.

LISTINGUL 16.11. Un script care extrage date şi afişează rezultatele.

```
<?php
// deschide conexiunea
$conn = mysql_connect("localhost", "joeuser", "somepass");
// alege baza de date care va fi folosită
mysql_select_db("testDB",$conn);
// compune instrucţiunea SQL
$sql = "SELECT * FROM testTable";
// execută instrucţiunea SQL
$result = mysql_query($sql, $conn) or die(mysql_error());
// parcuge fiecare linie din setul de rezultate si afiseaza datele
while ($newArray = mysql_fetch_array($result)) {
    // da un nume campurilor
    $id  = $newArray['id'];
    $testField = $newArray['testField'];
    // afiseaza pe ecran rezultatele
    echo "The ID is $id and the text is $testField <br>";
}
?>
```

Salvaţi acest script sub numele `select.php` şi puneţi-l în zona de documente a serverului dumneavoastră Web şi accesaţi-l în browser. Ar trebui să vedeţi un mesaj pentru fiecare înregistrare din tabelă, aşa cum este în figura 16.4.

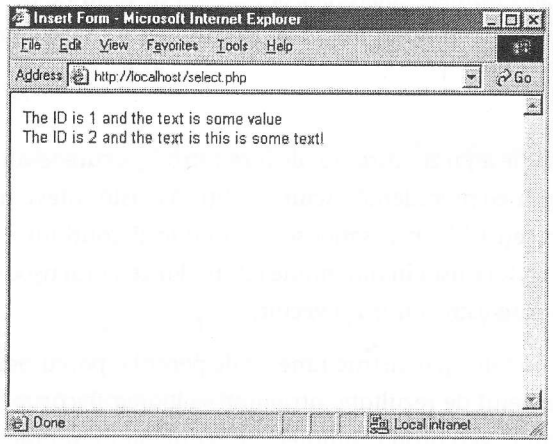


FIGURA 16.4. Selectarea înregistrărilor din MySQL.

Practic, puteţi crea o întreagă aplicaţie de baze de date folosind doar patru sau cinci funcţii MySQL. Acest capitol abia dacă a atins câteva chestiuni superficiale despre folosirea în comun a scripturilor PHP şi a serverului MySQL; există multe alte funcţii de lucru cu MySQL în PHP.

Alte funcții PHP de lucru cu MySQL

Există aproximativ 40 de funcţii specifice MySQL în PHP. Multe dintre aceste funcţii reprezintă doar metode alternative de extragere a datelor sau sunt folosite pentru a afla informaţii despre structura tabelei respective.

Pentru o listă completă a funcţiilor acestora, împreună cu exemple practice, vizitaţi secţiunea dedicată MySQL-ului din Manualul de PHP la adresa `http://www.php.net/mysql`.

Rezumat

Crearea de site-uri Web dinamice, folosind PHP împreună cu MySQL, este floare la ureche. Trebuie să reţineţi doar că funcţiile PHP reprezintă, practic, o poartă către serverul de baze de date; orice aţi putea introduce în MySQL folosind interfaţa în linie de comandă, puteţi să trimiteţi folosind funcţia `mysql_query()`.

Pentru a conecta scriptul PHP la serverul MySQL, trebuie să ştiţi numele de utilizator de MySQL, parola şi numele bazei de date. Folosind funcţiile `mysql_connect()` şi `mysql_select_db()` puteţi să vă conectaţi şi să selectaţi o bază de date pe care să o folosiţi pe tot parcursul scriptului respectiv.

O dată conectat, puteţi trimite comenzi SQL standard prin intermediul funcţiei `mysql_query()`. Dacă aţi trimis o comandă `SELECT`, puteţi folosi `mysql_numrows()` pentru a număra înregistrările returnate în setul de rezultate. Dacă vreţi să afişaţi datele extrase, puteţi folosi `mysql_fetch_array()` pentru a prelua toate rezultatele în cadrul unui ciclu iterativ şi a le afişa pe ecran.

Atelier

Atelierul este conceput să anticipeze posibile întrebări, să vă ajute să recapitulaţi cele învăţate şi să începeţi să vă puneţi cunoştinţele în practică.

Test

1. Care este funcţia principală folosită pentru a realiza o conexiune între un script PHP şi serverul MySQL şi ce informaţii sunt necesare?
2. Ce funcţie PHP preia un mesaj de eroare MySQL?
3. Ce funcţie PHP este folosită pentru numărarea înregistrărilor dintr-un set de rezultate?

Soluţii

1. Funcţia `mysql_connect()` creează o conexiune cu serverul MySQL şi are nevoie de numele calculatorului gazdă, de numele de utilizator şi de parolă.
2. Funcţia `mysql_error()` returnează un mesaj de eroare MySQL.
3. Funcţia `mysql_numrows()` numără înregistrările dintr-un set de rezultate.

PARTEA a V-a

Proiecte fundamentale

Capitolul 17. Gestionarea unei liste de e-mail simple 335

Capitolul 18. Crearea unei agende de adrese online 347

Capitolul 19. Crearea unui forum de discuții simplu 371

Capitolul 20. Crearea interfeței pentru un magazin virtual 387

Capitolul 21. Crearea unui mecanism pentru un coş de cumpărături virtual 399

Capitolul 22. Crearea unui calendar simplu 413

Capitolul 23. Restricționarea accesului la aplicațiile dumneavoastră 427

Capitolul 24. Înregistrarea şi monitorizarea activității serverului Web 499

Capitolul 24. Localizarea aplicațiilor 467

CAPITOLUL 17
Gestionarea unei liste de e-mail simple

Acest capitol vă oferă, pentru prima oară, câteva mici proiecte practice concepute astfel încât să vă fixeze cunoştinţele despre folosirea PHP şi MySQL. În acest capitol veţi învăţa o metodă de creare a unei liste de distribuţie de mesaje e-mail, care poate fi folosită pentru trimiterea unor buletine informative sau a oricăror altor mesaje pe care doriţi să le trimiteţi către o listă de adrese de e-mail stocată într-o bază de date.

Mecanismul de expediere a mesajelor e-mail pe care îl veţi folosi în acest capitol nu se vrea un înlocuitor pentru aplicaţiile de gestiune a listelor de e-mail, care sunt concepute special pentru expedierea de mesaje în masă. Sistemul pe care îl veţi construi în cadrul acestei lecţii ar trebui folosit doar pentru liste mici, cu mai puţin de câteva sute de adrese de e-mail.

În acest capitol, veţi învăţa cum să:

- creaţi un formular şi un script de abonare/dezabonare;
- creaţi o interfaţă pentru trimiterea mesajelor dumneavoastră;
- creaţi scriptul care trimite mesajele dumneavoastră.

Dezvoltarea mecanismului de abonare

Aţi învăţat, în lecţiile precedente, că planificarea este cel mai important aspect al creării unui produs. În acest caz, trebuie să vă gândiţi la elementele de care veţi avea nevoie pentru mecanismul de abonare:

- o tabelă care să stocheze adresa de e-mail;
- o modalitate prin care utilizatorii să poată să îşi introducă şi să îşi elimine adresa de e-mail;
- un formular şi un script pentru trimiterea mesajelor.

Următoarele secţiuni vor descrie fiecare dintre aceste elemente separat.

Crearea tabelei `subscribers`

Teoretic, nu aveţi nevoie decât de un singur câmp în tabela `subscribers` (abonaţi), în care să stocaţi adresa de e-mail a utilizatorului. Cu toate acestea, ar trebui să aveţi şi un câmp `ID` astfel încât să existe o coerenţă între tabele, dar şi pentru că este mult mai simplu, în clauzele `where`, să vă referiţi la o înregistrare prin intermediul `ID`-ului decât folosind o adresă de e-mail lungă. De aceea, în acest caz, interogarea dumneavoastră MySQL va arăta în felul următor:

```
mysql> create table subscribers (
    -> id int not null primary key auto_increment,
    -> email varchar (150) unique not null
    -> );
Query OK, 0 rows affected (0.00 sec)
```

Observaţi folosirea clauzei `unique` în definiţia câmpului `email`. Acest lucru înseamnă că, deşi `id` este cheia primară, nici pentru câmpul `email` nu trebuie admise valori duplicate. Câmpul `email` este o cheie unică, iar câmpul `id` este cheia primară.

```
mysql> desc subscribers;
+-------+--------------+------+-----+---------+----------------+
| Field | Type         | Null | Key | Default | Extra          |
+-------+--------------+------+-----+---------+----------------+
| id    | int(11)      |      | PRI | NULL    | auto_increment |
| email | varchar(150) |      | UNI |         |                |
+-------+--------------+------+-----+---------+----------------+
2 rows in set (0.00 sec)
```

Acum că aveţi o tabelă, puteţi crea formularul şi scriptul care să pună valori în ea.

Crearea formularului de abonare

Formularul de abonare va fi, de fapt, un formular multifuncţional şi un script în acelaşi timp şi se va numi `manage.php`, iar el va trata atât cererile de abonare, cât şi cele de dezabonare. Listingul 17.1 arată codul scriptului `manage.php` care foloseşte câteva funcţii definite de utilizator pentru a elimina porţiunile de cod repetitiv.

LISTINGUL 17.1. Abonaţi-vă şi dezabonaţi-vă folosind `manage.php`.

```
1: <?php
2: //defineste cateva functii
3: function doDB() {
4:     global $conn;
5:    //se conecteaza la server si selecteaza baza de date; s-ar putea sa fie util
6:    $conn = mysql_connect("localhost", "joeuser", "somepass")
7:          or die(mysql_error());
8:    mysql_select_db("testDB",$conn)  or die(mysql_error());
9: }
10:
11: function emailChecker($email) {
12:     global $conn, $check_result;
```

LISTINGUL 17.1. (*continuare*)

```
13:     //verifica daca adresa email nu este deja in lista
14:     $check = "select id from subscribers where email = '$email'";
15:     $check_result = mysql_query($check,$conn) or die(mysql_error());
16: }
17:
18: //decide daca este nevoie sa afiseze formularul sau nu
19: if ($_POST[op] != "ds") {
20:     //in caz afirmativ, se creaza formularul
21:     $display_block = "
22:     <form method=POST action=\"$_SERVER[PHP_SELF]\">
23:
24:     <p><strong>Your E-Mail Address:</strong><br>
25:     <input type=text name=\"email\" size=40 maxlength=150>
26:
27:     <p><strong>Action:</strong><br>
28:     <input type=radio name=\"action\" value=\"sub\" checked> subscribe
29:     <input type=radio name=\"action\" value=\"unsub\"> unsubscribe
30:
31:     <input type=\"hidden\" name=\"op\" value=\"ds\">
32:
33:     <p><input type=submit name=\"submit\" value=\"Submit Form\"></p>
34:     </form>";
35:
36: } else if (($_POST[op] == "ds") && ($_POST[action] == "sub")) {
37:     //se incearca abonarea; se valideaza adresa email
38:     if ($_POST[email] == "") {
39:         header("Location: manage.php");
40:         exit;
41:     }
42:     //conectarea la baza de date
43:     doDB();
44:     //se verifica daca adresa email este in lista
45:     emailChecker($_POST[email]);
46:
47:     //se preia numarul de rezultate, si se trece la actiune
48:     if (mysql_num_rows($check_result) < 1) {
49:         //se adauga inregistrarea
50:         $sql = "insert into subscribers values('', '$_POST[email]')";
51:         $result = mysql_query($sql,$conn) or die(mysql_error());
52:         $display_block = "<P>Thanks for signing up!</P>";
53:     } else {
54:         //se afiseaza mesajul de eroare
55:         $display_block = "<P>You're already subscribed!</P>";
56:     }
57: } else if (($_POST[op] == "ds") && ($_POST[action] == "unsub")) {
58:     //se incearca dezabonarea; se valideaza adresa email
59:     if ($_POST[email] == "") {
60:     header("Location: manage.php");
61:         exit;
62:     }
63:     //conectare la baza de date
64:     doDB();
```

LISTINGUL 17.1. *(continuare)*

```
65:     //se verifica daca adresa email este in lista
66:     emailChecker($_POST[email]);
67:
68:     //se preia numarul de rezultate, si se trece la actiune
69:     if (mysql_num_rows($check_result) < 1) {
70:         //se afiseaza mesajul de eroare
71:         $display_block = "<P>Couldn't find your address!</P>
72:         <P>No action was taken.</P>";
73:     } else {
74:         //adresa este dezabonata
75:         $id = mysql_result($check_result, 0, "id");
76:         $sql = "delete from subscribers where id = '$id'";
77:         $result = mysql_query($sql,$conn) or die(mysql_error());
78:         $display_block = "<P>You're unsubscribed!</p>";
79:     }
80: }
81: ?>
82: <HTML>
83: <HEAD>
84: <TITLE>Subscribe/Unsubscribe</TITLE>
85: </HEAD>
86: <BODY>
87: <h1>Subscribe/Unsubscribe</h1>
88: <?php echo "$display_block"; ?>
89: </BODY>
90: </HTML>
```

Poate că listingul 17.1 este cam lung, dar nu este complicat. De fapt, ar fi putut fi şi mai lung dacă nu am fi definit acele funcţii la începutul scriptului. Ar trebui să vă definiţi propriile funcţii atunci când ştiţi că veţi refolosi anumite porţiuni de cod şi nu vreţi să le tastaţi de fiecare dată. Liniile 3–9 construiesc prima funcţie, `doDB()`, care stabileşte conexiunea cu baza de date aşa cum aţi învăţat în lecţiile precedente. Liniile 11–6 definesc o funcţie denumită `emailChecker()`, care prea datele introduse de utilizator şi returnează o valoare – la fel ca majoritatea funcţiilor. Vom analiza această funcţie în contextul scriptului, atunci când ajungem la ea.

Pe linia 19 începe porţiunea principală a scriptului. Deoarece acest script efectuează mai multe acţiuni, trebuie să stabilim pe care dintre ele trebuie să o încerce în momentul de faţă. Dacă valoarea lui `$_POST[op]` nu este `"ds"`, ştim că utilizatorul nu a transmis formularul; de aceea, trebuie să îl afişăm pe acesta utilizatorului. Liniile 21–34 creează formularul de abonare / dezabonare, folosind acţiunea `$_SERVER[PHP_SELF]` (linia 22), creând un câmp denumit `email` pentru adresa utilizatorului şi două butoane radio (liniile 28–29) pentru a cu ajutorul cărora va fi introdusă acţiunea dorită. În acest punct, scriptul întrerupe blocul de cod `if...else`, sare direct la linia 82 şi trece la afişarea codului HTML. Formularul afişat seamănă cu cel din figura 17.1.

FIGURA 17.1. Formularul de abonare / dezabonare.

Dacă, însă, valoarea lui `$_POST[op]` este într-adevăr `"ds"`, trebuie să facem ceva. Avem două posibilităţi: abonare sau dezabonare. Stabilim ce tip de acţiune să efectuăm pe baza valorii din `$_POST[action]` – grupul de butoane radio.

Pe linia 36, dacă `$_POST[op]` este `"ds"`, iar `$_POST[action]` este `"sub"`, ştim că utilizatorul încearcă să se aboneze. Pentru a se abona, el trebuie să trimită o adresă de e-mail, aşa că verificăm dacă a făcut asta pe liniile 38–41. Dacă nu este prezentă nici o adresă, atunci utilizatorul este trimis înapoi la formular.

Dacă adresa este prezentă, atunci apelăm funcţia `doDB()` pe linia 43 pentru a ne conecta la baza de date fiindcă trebuie să executăm o interogare (sau două). Pe linia 45, apelăm a doua funcţie definită de utilizator: `emailChecker()`. Această funcţie preia datele introduse (`$_POST[email]`) şi le procesează. Dacă ne uităm sus, pe liniile 12–15, vedem că funcţia verifică dacă există o valoare `id` în tabela `subscribers` care să corespundă valorii introduse de utilizator. Funcţia returnează, apoi, setul de rezultate în variabila `$check_result` pentru ca acesta să poată fi folosit de restul scriptului.

> Observaţi definirea variabilelor globale la începutul ambelor funcţii definite de utilizator în listingul 17.1. Aceste variabile trebuie să fie partajate cu întregul script, aşa că este folosită directiva global la definirea lor.

Apropo

Uitaţi-vă acum la linia 48 pentru a vedea cum este folosită variabila `$check_result`: numărul de înregistrări din setul de rezultate referit de `$check_result` este mai mic decât 1, înseamnă că adresa nu este în listă, aşa că poate fi adăugată. Înregistrarea este adăugată, iar răspunsul este stocat pe liniile 50–52, iar mesajul de eroare (în cazul în care adresa este deja în tabelă) este stocat pe linia 55.

În acest moment, scriptul întrerupe blocul de cod `if...else`, sare direct la linia 82, şi începe să afişeze codul HTML. Veţi testa această funcţionalitate mai târziu.

Ultima variantă de date introduse de utilizator este atunci când valoarea lui `$_POST[op]` este `"ds"`, iar cea a lui `$_POST[action]` este `"unsub"`. În acest caz, utilizatorul încearcă să se dezaboneze. Pentru a se dezabona, el trebuie să introducă o adresă de e-mail, aşa că verificăm dacă a făcut asta pe liniile 59–61. Dacă nici o adresă nu este prezentă, utilizatorul este trimis înapoi la formular.

Dacă adresa este prezentă, apelăm funcţia `doDB()` pe linia 64 pentru a ne conecta la baza de date. Apoi, pe linia 66, apelăm funcţia `emailChecker()`, aceasta returnează din nou setul de rezultate, `$check_result`. Numărul de înregistrări este aflat pe linia 69, pentru a se stabili dacă adresa de e-mail este deja prezentă în tabelă. Dacă numărul este mai mic de 1, atunci adresa nu este în listă şi, deci, utilizatorul nu se poate dezabona. În acest caz, pe liniile 71–72, stocăm mesajul de eroare. Altminteri, utilizatorul este dezabonat (înregistrarea din tabelă este ştearsă) pe liniile 75–77 şi, pe linia 78, stocăm mesajul prin care îl anunţăm pe utilizator că a fost dezabonat cu succes. În acest moment, scriptul întrerupe blocul de cod `if...else`, sare direct la linia 82 şi începe să afişeze codul HTML.

Figurile de la 17.2 la 17.5 arată diversele rezultate ale scriptului, în funcţie de acţiunile selectate şi de prezenţa sau absenţa adresei de e-mail în baza de date.

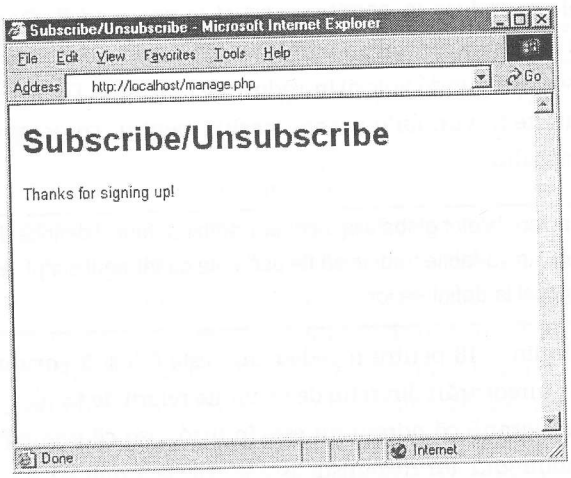


FIGURA 17.2. Acţiunea de abonare reuşită.

FIGURA 17.3. Acțiunea de abonare eşuată.

FIGURA 17.4. Acțiunea de dezabonare reuşită.

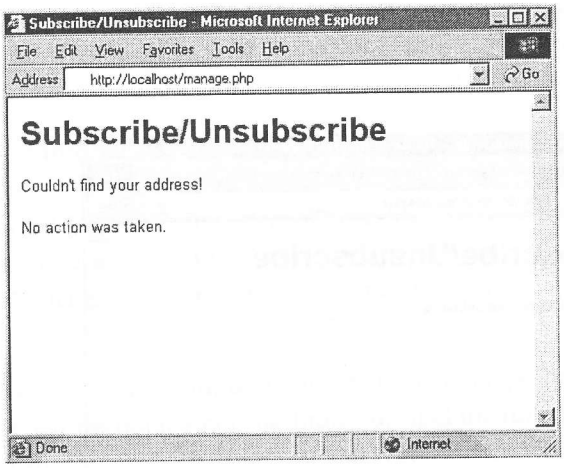


FIGURA 17.5. Acțiunea de dezabonare eșuată.

Mai departe, veți crea formularul și scriptul care trimite mesajele e-mail către toate adresele abonaților dumneavoastră.

Dezvoltarea mecanismului de trimitere a mesajelor e-mail

Având un mecanism de abonare pus la punct, puteți crea o interfață simplă pentru un script care va prelua conținutul unui formular și îl va trimite către toate adresele de e-mail din tabela `subscribers`. Acesta este un alt script din acelea multifuncționale, se numește `sendmymail.php` și se găsește în listingul 17.2.

LISTINGUL 17.2. Trimiterea mesajelor e-mail către lista de abonați.

```
1: <?php
2: if ($_POST[op] != "send") {
3:     //nu au vazut inca formularul, asa ca il afisam
4:     print "
5:     <HTML>
6:     <HEAD>
7:     <TITLE>Send a Newsletter</TITLE>
8:     </HEAD>
9:     <BODY>
10:    <h1>Send a Newsletter</h1>
11:    <form method=\"post\" action=\"$_SERVER[PHP_SELF]\">
12:    <P><strong>Subject:</strong><br>
13:    <input type=\"text\" name=\"subject\" size=30></p>
14:    <P><strong>Mail Body:</strong><br>
15:    <textarea name=\"message\" cols=50 rows=10 wrap=virtual></textarea>
16:    <input type=\"hidden\" name=\"op\" value=\"send\">
17:    <p><input type=\"submit\" name=\"submit\" value=\"Send It\"></p>
18:    </FORM>
```

LISTINGUL 17.2. (*continuare*)

```
19:     </BODY>
20:     </HTML>";
21:
22: } else if ($_POST[op] == "send") {
23:      //utilizatorul doreste sa trimita formularul; verificam campurile obligatori
24:      if (($_POST[subject] =="") || ($_POST[message] == "")) {
25:         header("Location: sendmymail.php");
26:         exit;
27:     }
28:
29:     //conectare la baza de date
30:     $conn = mysql_connect("localhost", "joeuser", "somepass")
31:          or die(mysql_error());
32:     mysql_select_db("testDB",$conn)  or die(mysql_error());
33:
34:     //se preiau adresele email din lista de abonati
35:     $sql = "select email from subscribers";
36:     $result = mysql_query($sql,$conn) or die(mysql_error());
37:
38:     //se creaza un antet de mesaj email From:
39:     $headers = "From: Your Mailing List <you@yourdomain.com>";
40:
41:     //se parcurg rezultatele si se trimite mesajul
42:     while ($row = mysql_fetch_array($result)) {
43:         set_time_limit(0);
44:         $email = $row['email'];
45:         mail("$email", stripslashes($_POST[subject]),
46:              stripslashes($_POST[message]), $headers);
47:         print "newsletter sent to: $email<br>";
48:     }
49: }
50: ?>
```

Corpul principal al scriptului începe chiar pe linia 2. unde stabilim dacă utilizatorul a văzut deja formularul sau nu. Dacă valoarea lui `$_POST[op]` nu este `"send"`, ştim că utilizatorul nu a trimis formularul, aşa că trebuie să îl afişăm. Liniile 4–20 creează formularul de trimitere a buletinului informativ, care foloseşte `$_SERVER[PHP_SELF]` drept acţiune (linia 11), conţine un câmp de text numit `subject` pentru subiectul mesajului şi un element de tip `textarea` (zonă de introducere a textului) denumit `message` pentru corpul mesajului e-mail ce trebuie trimis. În acest punct, scriptul întrerupe blocul de cod `if...else` şi trece la afişarea codului HTML. Formularul afişat arată ca în figura 17.6.

Dacă valoarea lui `$_POST[op]` este cu adevărat `"send"`, trebuie să trimitem formularul către destinatari. Înainte să trimitem, trebuie să verificăm existenţa celor două elemente obligatorii: `$_POST[subject]` şi `$_POST[message]`. Dacă oricare dintre acestea două lipseşte, utilizatorul este trimis înapoi la formular.

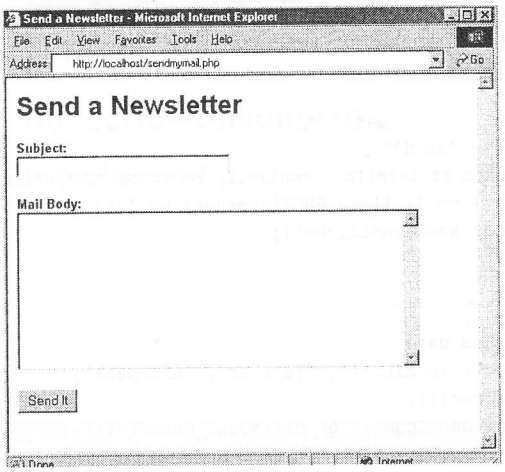


FIGURA 17.6. Formularul de trimitere a mesajului în masă.

Dacă elementele obligatorii sunt prezente, scriptul trece la liniile 30–32, care se conectează la baza de date. Interogarea este trimisă pe linia 36, care preia toate adresele de e-mail din tabela `subscribers`. Rezultatele nu sunt ordonate în nici un fel, deşi am putea să introducem o clauză `order by` dacă am dori să trimitem mesajele în ordine alfabetică.

Linia 39 creează un antet `From:`, care este folosit în bucla `while` imediat următoare. Acest antet asigură faptul că mesajul arată de parcă ar veni din partea unei persoane, nu a unei maşini. Bucla `while`, care începe pe linia 42, extrage fiecare adresă de e-mail din setul de rezultate. Pe linia 43, folosim funcţia `set_time_limit()` pentru a stabili limita de timp la valoarea `0`, care înseamnă „nelimitat". Acest lucru îi permite scriptului să ruleze atâta vreme cât are nevoie.

Apropo

Din cauză că scriptul din listingul 17.2 nu face altceva decât să execute funcţia `mail()` de foarte multe ori, el nu are o metodă de temporizare a trimiterii mesajelor, metodă indispensabilă aplicaţiilor reale de liste de e-mail, care are rostul de a nu supraîncărca serverul de mail. Folosirea funcţiei `set_time_limit()` nu reduce încărcarea serverului de mail; pur şi simplu, această funcţie îi dă voie scriptului să ruleze dincolo de momentul în care, altminteri, timpul său de execuţie ar fi expirat.

Mesajul e-mail este transmis pe linia 45 prin intermediul funcţiei `mail()`, inserând valorile din formular acolo unde trebuie. Linia 46 afişează pe ecran un mesaj care enumeră adresele către care a fost trimis mesajul. În figurile 17.7 şi 17.8 se pot vedea rezultatele scriptului.

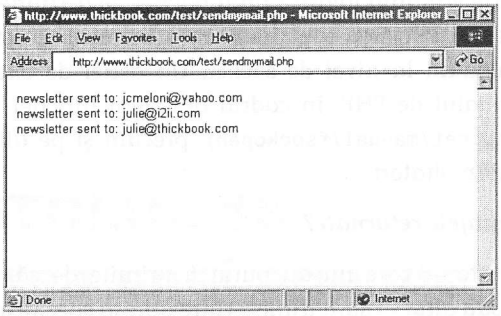


FIGURA 17.7. Mesajul a fost trimis!

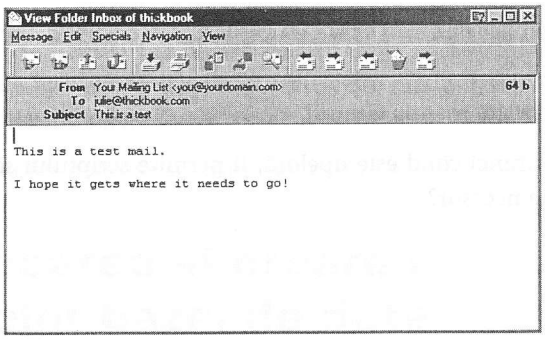


FIGURA 17.8. Mesajul a ajuns în siguranță.

Rezumat

În acest capitol axat pe practică, ați aplicat cunoștințele dumneavoastră de bază referitoare la PHP și MySQL pentru a crea o listă personală de expediție de mesaje e-mail. Printre activități s-au găsit crearea unei tabele, implementarea mecanismelor de abonare și dezabonare și realizarea formularului și a scriptului de trimitere a mesajelor e-mail.

Întrebări şi răspunsuri

Î. Cum aş putea preveni supraîncărcarea serverului de mail?

A. Lăsând la o parte varianta folosirii unui pachet de gestiune a listelor de e-mail, puteţi evita folosirea funcţiei `mail()` lucrând direct cu serverul SMTP prin conexiuni la nivel de socket. Un astfel de exemplu poate fi găsit în Manualul de PHP, în cadrul descrierii funcţiei `fsockopen()` (`http://www.php.net/manual/fsockopen`), precum şi pe alte site-uri cu resurse pentru dezvoltatori.

Î. Unde se duc mesajele returnate?

A. Mesajele returnate — care nu au putut fi primite de către destinatar dintr-un motiv sau altul — sunt trimise către adresa specificată în antetul `Reply-to:` sau `From:` al mesajului dumneavoastră.

Atelier

Atelierul este conceput să anticipeze posibile întrebări, să vă ajute să recapitulaţi cele învăţate şi să începeţi să vă puneţi cunoştinţele în practică.

Test

1. Ce funcţie trimite mesaje e-mail?
2. Ce funcţie, atunci când este apelată, îi permite scriptului să ruleze atât timp cât este necesar?

Soluţii

1. Nu este o întrebare capcană. Este vorba de funcţia `mail()`!
2. `set_time_limit(0)`

CAPITOLUL 18
Crearea unei agende de adrese online

În această lecţie practică, veţi crea o agendă de adrese (address book) care să poată fi gestionat online. Veţi învăţa metodele necesare pentru crearea tabelelor de bază de date, precum şi a formularelor şi scripturilor pentru adăugarea, ştergerea şi vizualizarea înregistrărilor din baza de date.

În acest capitol, veţi învăţa cum să:

- creaţi o bază de date relaţională pentru o agendă de adrese online;
- creaţi formularele şi scripturile pentru adăugarea şi ştergerea înregistrărilor din agendă;
- creaţi formularele şi scripturile pentru vizualizarea înregistrărilor.

Planificarea şi crearea tabelelor bazei de date

Când vă gândiţi la o agendă de adrese, vă vin în minte imediat câmpurile evidente: numele, adresa, numărul de telefon, adresa de e-mail. Cu toate acestea, dacă vă uitaţi la agenda dumneavoastră reală, veţi observa că aveţi mai multe intrări pentru aceeaşi persoană. Se poate ca o persoană să aibă trei numere de telefon, sau două adrese de e-mail şi aşa mai departe. În cazul agendei dumneavoastră online, un set de tabele relaţionate vor ameliora problema redundanţei şi a informaţiilor repetate.

Tabelul 18.1 arată un exemplu de tabelă cu nume de câmpuri ce ar putea fi folosite la crearea agendei dumneavoastră. În câteva momente, veţi crea şi instrucţiunile SQL efective, dar mai întâi ar trebui să vă uitaţi la aceste informaţii şi să înţelegeţi cum se structurează relaţiile. Întrebaţi-vă ce câmpuri ar trebui să fie chei primare sau unice.

TABELUL 18.1. Tabelele agendei de adrese şi numele câmpurilor.

Numele tabelei	Numele câmpurilor
master_name	id, date_added, date_modified, f_name, l_name
address	id, master_id, date_added, date_modified, address, city, state, zipcode, type
telephone	id, master_id, date_added, date_modified, tel_number, type
fax	id, master_id, date_added, date_modified, fax_number, type
email	id, master_id, date_added, date_modified, email, type
personal_notes	id, master_id, date_added, date_modified, note

Observaţi folosirea câmpurilor calendaristice; fiecare tabelă are un câmp `date_added` şi `date_modified` (data adăugării şi data modificării). Aceste câmpuri vor fi de ajutor la întreţinerea bazei de date; la un moment dat s-ar putea să rulaţi o interogare care să elimine toate înregistrările care sunt mai vechi de un număr oarecare de luni sau ani, sau care să elimine toate înregistrările care nu au mai fost actualizate de ceva vreme.

După cum puteţi vedea în următoarea instrucţiune SQL, tabela `master_name` (nume principal) are încă două câmpuri pe lângă ID şi cele calendaristice: `f_name` şi `l_name`, pentru prenume şi, respectiv, pentru numele de familie. Câmpul `id` este cheia primară. Nici o altă cheie nu trebuie să fie primară sau unică, în afara cazului în care vreţi ca în baza de date să nu existe decât un singur John Smith, o singură Mary Jones etc.

Apropo

> Lungimile câmpurilor de tip text din următoarele instrucţiuni sunt arbitrare; puteţi să le faceţi cât de lungi sau scurte doriţi, în limitele permise la definirea unui câmp.

```
mysql> create table master_name (
    -> id int not null primary key auto_increment,
    -> date_added datetime,
    -> date_modified datetime,
    -> f_name varchar (75),
    -> l_name varchar (75)
    -> );
Query OK, Ø rows affected (Ø.Ø1 sec)
```

Tabela `address` are cheia primară tipică `id`, precum şi câmpurile `date_added` şi `date_modified`. În plus, ar trebui să vă daţi acum seama unde va fi creată relaţia — prin intermediul câmpului `master_id`. Acest câmp va fi egal cu câmpul `id` din tabela `master_table` corespunzător persoanei căreia îi aparţine adresa. Câmpul `master_id` nu este o cheie unică pentru că putem presupune că o persoană ar putea avea mai multe adrese în acelaşi timp. Vedem acest lucru în câmpul `type`,

care este definit ca o enumerare ce conţine trei opţiuni: `home`, `work` şi `other` (acasă, la serviciu şi altele). O persoană poate avea mai multe adrese din fiecare tip, aşa că nu mai există alte chei în această tabelă în afară de `id`. Presupunând că această agendă conţine numai adrese din Statele Unite, am mai adăugat în tabelă şi câmpurile `address`, `city`, `state` şi `zipcode` (adresa, oraşul, statul şi codul ZIP — un fel de cod poştal folosit în Statele Unite).

```
mysql> create table address (
    -> id int not null primary key auto_increment,
    -> master_id int not null,
    -> date_added datetime,
    -> date_modified datetime,
    -> address varchar (255),
    -> city varchar (30),
    -> state char (2),
    -> zipcode varchar (10),
    -> type enum ('home', 'work', 'other')
    -> );
Query OK, 0 rows affected (0.01 sec)
```

Tabelele `telephone`, `fax` şi `email` sunt variaţii pe aceeaşi temă.

```
mysql> create table telephone (
    -> id int not null primary key auto_increment,
    -> master_id int not null,
    -> date_added datetime,
    -> date_modified datetime,
    -> tel_number varchar (25),
    -> type enum ('home', 'work', 'other')
    -> );
Query OK, 0 rows affected (0.01 sec)

mysql> create table fax (
    -> id int not null primary key auto_increment,
    -> master_id int not null,
    -> date_added datetime,
    -> date_modified datetime,
    -> fax_number varchar (25),
    -> type enum ('home', 'work', 'other')
    -> );
Query OK, 0 rows affected (0.00 sec)

mysql> create table email (
    -> id int not null primary key auto_increment,
    -> master_id int not null,
    -> date_added datetime,
    -> date_modified datetime,
    -> email varchar (150),
    -> type enum ('home', 'work', 'other')
    -> );
Query OK, 0 rows affected (0.00 sec)
```

Tabela `personal_notes` urmează acelaşi model, cu excepţia faptului că `master_id` este o cheie unică permiţând doar o singură înregistrare pentru fiecare persoană:

```
mysql> create table personal_notes (
    -> id int not null primary key auto_increment,
    -> master_id int not null unique,
    -> date_added datetime,
    -> date_modified datetime,
    -> note text
    -> );
Query OK, Ø rows affected (Ø.ØØ sec)
```

Acum că aţi creat tabelele, puteţi să lucraţi la formularele şi scripturile dedicate gestionării şi vizualizării înregistrărilor dumneavoastră.

Crearea unui meniu

Agenda dumneavoastră de adrese online va avea mai multe acţiuni, aşa că este logic să creaţi un meniu pentru acestea. Listingul 18.1 creează un meniu pentru fiecare dintre scripturile pe care le veţi scrie în această secţiune şi care se numeşte `mymenu.php`.

LISTINGUL 18.1. Meniul agendei de adrese.

```
 1: <html>
 2: <head>
 3: <title>My Address Book</title>
 4: </head>
 5: <body>
 6: <h1>My Address Book</h1>
 7:
 8: <P><strong>Management</strong>
 9: <ul>
1Ø: <li><a href="addentry.php">Add an Entry</a>
11: <li><a href="delentry.php">Delete an Entry</a>
12: </ul>
13:
14: <P><strong>Viewing</strong>
15: <ul>
16: <li><a href="selentry.php">Select a Record</a>
17: </ul>
18: </body>
19: </html>
```

Figura 18.1 arată rezultatul listingului 18.1. Veţi aborda fiecare dintre aceste elemente în ordine, începând, în secţiunea următoare, cu „Add an entry“ (adaugă o înregistrare).

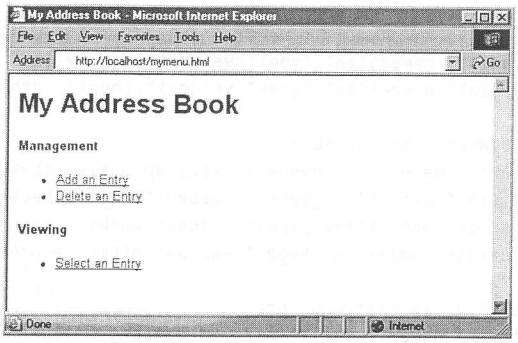


FIGURA 18.1. Meniul agendei de adrese.

Crearea mecanismului de adăugare a înregistrărilor

Doar pentru că veţi adăuga informaţie în şase tabele, nu înseamnă că formularul sau scriptul dumneavoastră va fi monstruos. De fapt, scripturile nu vor fi prea diferite de altele pe care le-aţi creat în cadrul lecţiilor anterioare şi, cu antrenament, veţi fi capabili să transformaţi aceste scripturi stufoase în versiuni mai cursive şi mai eficiente.

În listingul 18.2 puteţi vedea un script simplu de adăugare a unei înregistrări, pe care l-am denumit `addentry.php`.

LISTINGUL 18.2. Un script simplu de adăugare a unei înregistrări, denumit `addentry.php`.

```
<?php
if ($_POST[op] != "add") {
   //nu au vazut inca formularul, asa ca il afisam
   $display_block = "<h1>Add an Entry</h1>
   <form method=\"post\" action=\"$_SERVER[PHP_SELF]\">
   <P><strong>First/Last Names:</strong><br>
   <input type=\"text\" name=\"f_name\" size=30 maxlength=75>
   <input type=\"text\" name=\"l_name\" size=30 maxlength=75>

   <P><strong>Address:</strong><br>
   <input type=\"text\" name=\"address\" size=30>

   <P><strong>City/State/Zip:</strong><br>
   <input type=\"text\" name=\"city\" size=30 maxlength=50>
   <input type=\"text\" name=\"state\" size=5 maxlength=2>
   <input type=\"text\" name=\"zipcode\" size=10 maxlength=10>

   <P><strong>Address Type:</strong><br>
   <input type=\"radio\" name=\"add_type\" value=\"home\" checked> home
   <input type=\"radio\" name=\"add_type\" value=\"work\"> work
   <input type=\"radio\" name=\"add_type\" value=\"other\"> other

   <P><strong>Telephone Number:</strong><br>
   <input type=\"text\" name=\"tel_number\" size=30 maxlength=25>
   <input type=\"radio\" name=\"tel_type\" value=\"home\" checked> home
```

LISTINGUL 18.2. (*continuare*)

```
26:    <input type=\"radio\" name=\"tel_type\" value=\"work\"> work
27:    <input type=\"radio\" name=\"tel_type\" value=\"other\"> other
28:
29:    <P><strong>Fax Number:</strong><br>
30:    <input type=\"text\" name=\"fax_number\" size=30 maxlength=25>
31:    <input type=\"radio\" name=\"fax_type\" value=\"home\" checked> home
32:    <input type=\"radio\" name=\"fax_type\" value=\"work\"> work
33:    <input type=\"radio\" name=\"fax_type\" value=\"other\"> other
34:
35:    <P><strong>Email Address:</strong><br>
36:    <input type=\"text\" name=\"email\" size=30 maxlength=150>
37:    <input type=\"radio\" name=\"email_type\" value=\"home\" checked> home
38:    <input type=\"radio\" name=\"email_type\" value=\"work\"> work
39:    <input type=\"radio\" name=\"email_type\" value=\"other\"> other
40:
41:    <P><strong>Personal Note:</strong><br>
42:    <textarea name=\"note\" cols=35 rows=5 wrap=virtual></textarea>
43:     <input type=\"hidden\" name=\"op\" value=\"add\">
44:
45:    <p><input type=\"submit\" name=\"submit\" value=\"Add Entry\"></p>
46:    </FORM>";
47:
48: } else if ($_POST[op] == "add") {
49:     //este timpul sa adaugam date in tabele; verificam completarea campurilor
50:      if (($_POST[f_name] == "") || ($_POST[l_name] == "")) {
51:         header("Location: addentry.php");
52:         exit;
53:      }
54:
55:     //conectare la baza de date
56:     $conn = mysql_connect("localhost", "joeuser", "somepass")
57:         or die(mysql_error());
58:     mysql_select_db("testDB",$conn)  or die(mysql_error());
59:
60:     //adaugare la tabela master_name
61:     $add_master = "insert into master_name values ('', now(), now(),
62:         '$_POST[f_name]', '$_POST[l_name]')";
63:     mysql_query($add_master) or die(mysql_error());
64:
65:     //preia master_id pentru a-l utiliza cu alte tabele
66:     $master_id = mysql_insert_id();
67:
68:     if (($_POST[address]) || ($_POST[city]) || ($_POST[state]) ||
69:          ($_POST[zipcode])) {
70:         //informatii relevante, care se adauga la tabela address
71:         $add_address = "insert into address values ('', $master_id,
72:              now(), now(), '$_POST[address]', '$_POST[city]',
73:              '$_POST[state]', '$_POST[zipcode]', '$_POST[add_type]')";
74:         mysql_query($add_address) or die(mysql_error());
75:     }
76:
77:     if ($_POST[tel_number]) {
78     //informatii relevante, care se adauga la tabela telephone
79:         $add_tel = "insert into telephone values ('', $master_id,
```

LISTINGUL 18.2. (*continuare*)

```
80:          now(), now(), '$_POST[tel_number]',  '$_POST[tel_type]')";
81:         mysql_query($add_tel) or die(mysql_error());
82:     }
83:
84:     if ($_POST[fax_number]) {
85:         //informatii relevante, care se adauga la tabela fax
86:         $add_fax = "insert into fax values ('', $master_id, now(),
87:          now(), '$_POST[fax_number]',  '$_POST[fax_type]')";
88:         mysql_query($add_fax) or die(mysql_error());
89:     }
90:
91:     if ($_POST[email]) {
92:         //informatii relevante, care se adauga la tabela email
93:         $add_email = "insert into email values ('', $master_id,
94:              now(), now(), '$_POST[email]',  '$_POST[email_type]')";
95:         mysql_query($add_email) or die(mysql_error());
96:     }
97:
98:     if ($_POST[note]) {
99:         //informatii relevante, care se adauga la tabela notes
100:        $add_note = "insert into personal_notes values ('', $master_id,
101:             now(), now(), '$_POST[note]')";
102:        mysql_query($add_note) or die(mysql_error());
103:    }
104:
105:    $display_block = "<h1>Entry Added</h1>
106:    <P>Your entry has been added.  Would you like to
107:     <a href=\"addentry.php\">add another</a>?</p>";
108: }
109: ?>
110: <HTML>
111: <HEAD>
112: <TITLE>Add an Entry</TITLE>
113: </HEAD>
114: <BODY>
115: <?php echo $display_block; ?>
116: </BODY>
117: </HTML>
```

Acest script poate efectua două sarcini şi va o îndeplini pe una dintre ele atunci când este apelat: fie afişează formularul de adăugare a înregistrărilor, sau execută interogările SQL menite să adauge efectiv înregistrarea. Blocul de cod care stabileşte ce sarcină va fi îndeplinită începe pe linia 2, cu o verificare a valorii lui `$_POST[op]`. Dacă această valoare nu este `"add"`, atunci utilizatorul nu a văzut formularul şi, de aceea, formularul trebuie afişat. Codul HTML corespunzător formularului este amplasat într-un şir de caractere numit `$display_block`, de pe liniile 4–55. Apoi, scriptul întrerupe blocul de cod `if...else` şi sare la linia 110, care trece în modul de afişare a codului HTML, şi afişează valoarea variabilei `$display_block`, care, în cazul acesta, reprezintă formularul. Acest rezultat este ilustrat în figura 18.2.

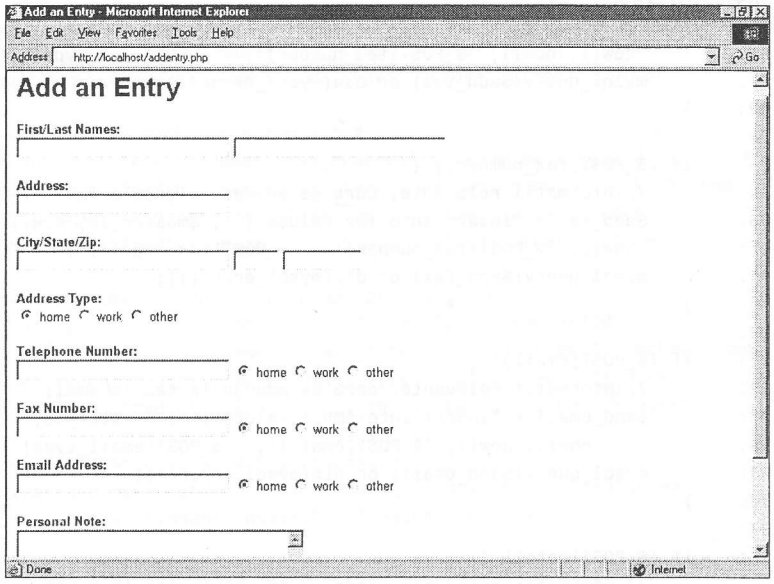


FIGURA 18.2. Formularul de adăugare a înregistrărilor.

Linia 48 începe codul ce trebuie executat în cea de-a doua condiţie, dacă valoarea lui `$_POST[op]` este `"add"`, ceea ce înseamnă că utilizatorul a trimis formularul. De dragul discuţiei, două dintre câmpuri au fost desemnate ca fiind obligatorii: prenumele şi numele de familie ale persoanei. De aceea, pe liniile 50–53 este verificată existenţa unor valori în `$_POST[f_name]` şi `$_POST[l_name]`, iar utilizatorul este trimis înapoi la formular dacă oricare dintre aceste valori lipseşte.

După ce ne-am asigurat că cele două câmpuri obligatorii există, ne conectăm la baza de date pe liniile 56–59. După aceasta urmează mulţimea de instrucţiuni de inserare, dintre care numai una singură este cu siguranţă necesară — inserarea unei înregistrări în tabela `master_name`. Acest lucru se petrece pe liniile 61–63. După ce este efectuată această inserare, extragem valoarea câmpului `id` a acestei înregistrări, folosind funcţia `mysql_insert_id()`, pe linia 66. Vom folosi această valoare, stocată acum în variabila `$master_id`, pentru construirea restului interogărilor SQL.

Interogările SQL pentru inserarea înregistrărilor în restul tabelelor sunt, toate, condiţionale. Acest lucru înseamnă că ele nu se execută decât dacă este verificată o anumită condiţie. Pe liniile 68–69, observăm că această condiţie ce trebuie îndeplinită este să existe valori în variabilele corespunzătoare: `$_POST[address]`, `$_POST[city]`, `$_POST[state]`, `$_POST[zipocode]`. Liniile 70–74 creează interogările şi le trimit dacă este îndeplinită condiţia respectivă.

Acelaşi principiu se aplică şi pentru adăugarea înregistrării în tabelele `telephone` (liniile 77–82), `fax` (liniile 84–89), `email` (liniile 91–96) şi `personal_notes` (liniile 98–103). O dată trecut de acest set de condiţii, scriptul continuă, mesajul pentru

utilizator fiind stocat în variabila `$display_block`, părăseşte blocul de cod `if... else` şi afişează codul HTML pe liniile 110–117.

Rezultatul afişat în urma adăugării unei înregistrări este ilustrat în figura 18.3.

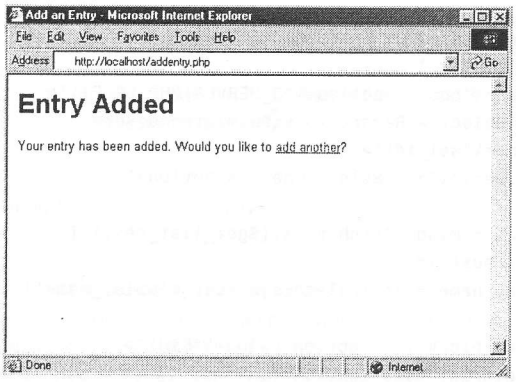


FIGURA 18.3. Adăugarea unei înregistrări.

Adăugaţi câteva înregistrări folosind acest formular astfel încât să aveţi câteva valori cu care să vă jucaţi în secţiunile următoare. Încercaţi să modificaţi scriptul acesta singuri în aşa fel încât valorile introduse în formular să fie afişate pe ecran după inserarea înregistrării.

Vizualizarea înregistrărilor

Dacă aţi verificat operaţiunile din secţiunea precedentă rulând interogări prin intermediul monitorului MySQL sau a altei interfeţe, probabil că v-aţi săturat să tastaţi `SELECT * FROM...` pentru fiecare tabel. În această secţiune, veţi crea un script cu două părţi care vă arată cum să selectaţi şi să vizualizaţi înregistrările din baza dumneavoastră de date.

LISTINGUL 18.3. Un script denumit `selentry.php` pentru selectarea şi vizualizarea înregistrărilor.

```
 1: <?php
 2: //conectare la baza de date
 3: $conn = mysql_connect("localhost", "joeuser", "somepass")
 4:    or die(mysql_error());
 5: mysql_select_db("testDB",$conn)  or die(mysql_error());
 6:
 7: if ($_POST[op] != "view")  {
 8:    //nu au vazut inca formularul, asa ca il afisam
 9:    $display_block = "<h1>Select an Entry</h1>";
10:
11:    //preia o parte din inregistrari
12:    $get_list = "select id, concat_ws(', ', l_name, f_name) as display_name
13:        from master_name order by l_name, f_name";
14:    $get_list_res = mysql_query($get_list) or die(mysql_error());
15:
```

LISTINGUL 18.3. (*continuare*)

```
16:    if (mysql_num_rows($get_list_res) < 1) {
17:        //nici o inregistrare
18:        $display_block .= "<p><em>Sorry, no records to select!</em></p>";
19:
20:    } else {
21:        //exista inregistrari; preia rezultatele si le afiseaza in formular
22:        $display_block .= "
23:        <form method=\"post\" action=\"$_SERVER[PHP_SELF]\">
24:        <P><strong>Select a Record to View:</strong><br>
25:        <select name=\"sel_id\">
26:        <option value=\"\">-- Select One --</option>";
27:
28:        while ($recs = mysql_fetch_array($get_list_res)) {
29:            $id = $recs['id'];
30:            $display_name = stripslashes($recs['display_name']);
31:
32:            $display_block .= "<option value=\"$id\">
33:                 $display_name</option>";
34:        }
35:        $display_block .= "
36:        </select>
37:        <input type=\"hidden\" name=\"op\" value=\"view\">
38:        <p><input type=\"submit\" name=\"submit\"
39:            value=\"View Selected Entry\"></p>
40:        </FORM>";
41:    }
42:
43:} else if ($_POST[op] == "view") {
44:
45:    //verificam completarea campurilor obligatorii
46:     if ($_POST[sel_id] == "")  {
47:        header("Location: selentry.php");
48:        exit;
49:    }
50:
51:    //preia master_info
52:    $get_master = "select concat_ws(' ', f_name, l_name) as display_name
53:         from master_name where id = $_POST[sel_id]";
54:    $get_master_res = mysql_query($get_master);
55:    $display_name = stripslashes(mysql_result($get_master_res,
56:         0,'display_name'));
57:    $display_block = "<h1>Showing Record for $display_name</h1>";
58:    //preia toate adresele
59:    $get_addresses = "select address, city, state, zipcode, type
60:         from address where master_id = $_POST[sel_id]";
61:    $get_addresses_res = mysql_query($get_addresses);
62:
63:    if (mysql_num_rows($get_addresses_res) > 0) {
64:
65:        $display_block .= "<P><strong>Addresses:</strong><br>
66:        <ul>";
67:
```

LISTINGUL 18.3. (*continuare*)

```
68:         while ($add_info = mysql_fetch_array($get_addresses_res)) {
69:             $address = $add_info[address];
70:             $city = $add_info[city];
71:             $state = $add_info[state];
72:             $zipcode = $add_info[zipcode];
73:             $address_type = $add_info[type];
74:
75:             $display_block .= "<li>$address $city $state $zipcode
76:                 ($address_type)";
77:         }
78:
79:         $display_block .= "</ul>";
80:     }
81:
82:     //preia toate numerele de telefon
83:     $get_tel = "select tel_number, type from telephone where
84:          master_id = $_POST[sel_id]";
85:     $get_tel_res = mysql_query($get_tel);
86:
87:     if (mysql_num_rows($get_tel_res) > 0) {
88:
89:         $display_block .= "<P><strong>Telephone:</strong><br>
90:         <ul>";
91:
92:         while ($tel_info = mysql_fetch_array($get_tel_res)) {
93:             $tel_number = $tel_info[tel_number];
94:             $tel_type = $tel_info[type];
95:
96:             $display_block .= "<li>$tel_number ($tel_type)";
97:         }
98:
99:         $display_block .= "</ul>";
100:    }
101:
102:    //preia toate numerele de fax
103:    $get_fax = "select fax_number, type from fax where
104:         master_id = $_POST[sel_id]";
105:    $get_fax_res = mysql_query($get_fax);
106:
107:    if (mysql_num_rows($get_fax_res) > 0) {
108:
109:        $display_block .= "<P><strong>Fax:</strong><br>
110:        <ul>";
111:
112:        while ($fax_info = mysql_fetch_array($get_fax_res)) {
113:            $fax_number = $fax_info[fax_number];
114:            $fax_type = $fax_info[type];
115:
116:            $display_block .= "<li>$fax_number ($fax_type)";
117:        }
118:
119:        $display_block .= "</ul>";
```

LISTINGUL 18.3. (*continuare*)

```
120:    }
121:
122:    //preia toate adresele email
123:    $get_email = "select email, type from email where
124:         master_id = $_POST[sel_id]";
125:    $get_email_res = mysql_query($get_email);
126:
127:    if (mysql_num_rows($get_email_res) > Ø) {
128:
129:        $display_block .= "<P><strong>Email:</strong><br>
130:        <ul>";
131:
132:        while ($email_info = mysql_fetch_array($get_email_res)) {
133:            $email = $email_info[email];
134:            $email_type = $email_info[type];
135:
136:            $display_block .= "<li>$email ($email_type)";
137:        }
138:
139:        $display_block .= "</ul>";
140:    }
141:
142:    //preia informatiile suplimentare
143:    $get_notes = "select note from personal_notes where
144:         master_id = $_POST[sel_id]";
145:    $get_notes_res = mysql_query($get_notes);
146:
147:    if (mysql_num_rows($get_notes_res) == 1) {
148:        $note = nl2br(stripslashes(mysql_result($get_notes_res,Ø,'note')));
149:
150:        $display_block .= "<P><strong>Personal Notes:</strong><br>$note";
151:    }
152:
153:    $display_block .= "<br><br><P align=center>
154:         <a href=\"$_SERVER[PHP_SELF]\">select another</a></p>";
155: }
156: ?>
157: <HTML>
158: <HEAD>
159: <TITLE>My Records</TITLE>
160: </HEAD>
161: <BODY>
162: <?php echo $display_block; ?>
163: </BODY>
164: </HTML>
```

La fel ca în cazul scriptului `addentry.php`, scriptul `selentry.php` va efectua una dintre cele două sarcini pe care le poate îndeplini: fie afişează formularul de selecţie, fie rulează toate interogările SQL necesare pentru vizualizarea înregistrărilor. Indiferent pe care dintre sarcini o efectuează, scriptul va trebui să folosească baza de date. Dat fiind acest fapt, ne conectăm la baza de date încă de pe liniile 3–5.

Porţiunea de cod care determină operaţiunea ce trebuie efectuată începe pe linia 7, testând valoarea lui `$_POST[op]`. Dacă valoarea lui `$_POST[op]` nu este `"view"`, utilizator nu vine dinspre formularul nostru şi, de aceea, trebuie să îi afişăm acest formular. Pe linia 9 începe construirea unui şir denumit `$display_block`, iar acest şir va fi completat pe parcursul operaţiunii. Speranţa noastră este că acest şir va conţine, în final, un formular de selecţie.

Pe liniile 12–14 selectăm o parte dintre înregistrările din tabela `master_name` pentru a construi opţiunile formularului de selecţie. Pentru primul pas, avem nevoie doar de numele şi ID-ul persoanei a cărei înregistrări o selectăm. Pe linia 16 sunt testate rezultatele interogării. Dacă interogarea nu are nici un rezultat, nu puteţi construi formularul. În acest caz, valoarea lui `$display_block` va fi un mesaj de eroare, iar scriptul îşi va opri execuţia, afişând codul HTML rezultat până acum.

Pe de altă parte, presupunând că există câteva înregistrări în tabela `master_name`, trebuie să extragem rezultatele interogării pentru a putea construi formularul. Acest lucru este realizat pe liniile 28–33, iar elementele formularului sunt scrise în variabila `$display_block`, atât deasupra cât şi dedesubtul acestor linii. Apoi, scriptul părăseşte blocul de cod `if...else` şi sare la linia 110, care afişează codul HTML şi valoarea lui `$display_block`, în acest caz — formularul. Acest rezultat poate fi văzut în figura 18.4.

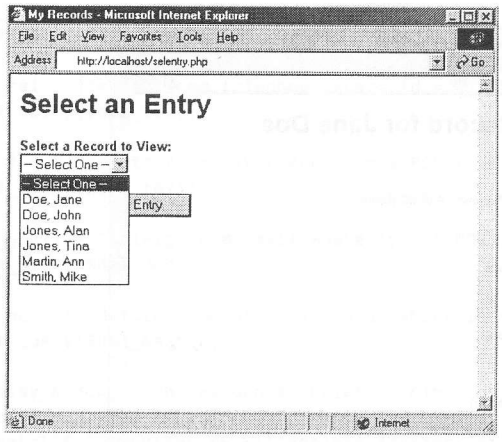


FIGURA 18.4. Formularul de selecţie a înregistrărilor.

Pe linia 43 începe porţiunea de cod ce se execută în cea de-a doua condiţie, adică atunci când valoarea lui `$_POST[op]` este `"view"`, ceea ce înseamnă că utilizatorul a transmis formularul şi vrea să vadă detaliile unei înregistrări anume. Câmpul obligatoriu pentru această secţiune a scriptului este `$_POST[sel_id]`, care conţine ID-ul din tabela `master_name` pe care utilizatorul l-a selectat în formularul de selecţie. Dacă această valoare nu există, utilizatorul este trimis înapoi la formularul de selecţie. Pe liniile 52–55, o interogare obţine numele persoanei a

cărei înregistrare vrea să o vadă utilizatorul. Această informaţie este amplasată în şirul de caractere `$display_block`, cu care, acum, sunteţi familiarizat şi care va continua să fie construit pe măsură ce scriptul îşi continuă execuţia.

Liniile 59–80 reprezintă interogările efectuate asupra tabelei `address`. Dacă persoana selectată nu are nici o înregistrare în tabela `address`, nu este adăugat nimic în şirul de caractere `$display_block`. Dacă, din contră, există mai multe înregistrări, acestea sunt scrise în `$display_block` sub forma unei liste cu marcatori, pe liniile 65–79.

Acelaşi principiu este aplicat şi pentru înregistrările din tabelele `telephone` (liniile 83–100), `fax` (liniile 103–120) şi `email` (liniile 123–140). Dacă există mai multe intrări, acestea sunt scrise în variabila `$display_block`. În caz contrar, scriptul trece mai departe. Deoarece, în tabela `personal_notes`, nu poate exista decât o singură înregistrare pentru fiecare persoană, scriptul verifică existenţa acesteia pe linia 143 şi trece mai departe dacă nu aceasta există. Dacă există, totuşi, o notă, ea este scrisă în variabila `$display_block` pe liniile 147–151.

Acţiunea finală a scriptului este afişarea unui link, pe liniile 153–154, pe care utilizatorul îl poate folosi în caz că doreşte să se întoarcă la formularul de selecţie. După acest punct, scriptul părăseşte blocul de cod `if...else` şi afişează codul HTML pe ecran. Figura 18.5 arată o înregistrare afişată de scriptul de selecţie a înregistrărilor, având câte o înregistrare în fiecare dintre tabele.

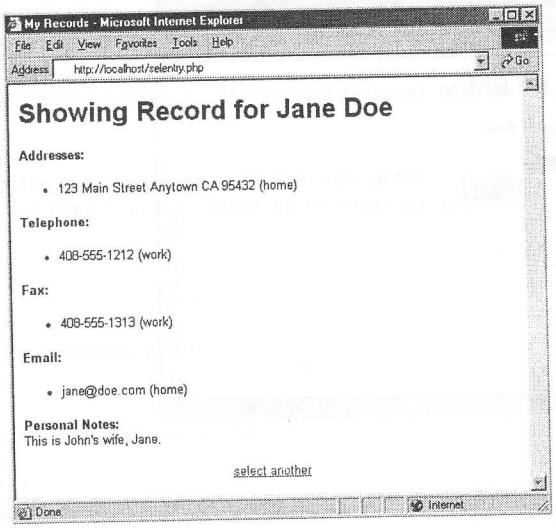


FIGURA 18.5. Înregistrarea corespunzătoare unei persoane.

Încercaţi şi dumneavoastră acest script. Ar trebui să vedeţi datele numai pentru persoanele care au date introduse în baza de date. De exemplu, dacă aveţi o înregistrare a unui prieten şi tot ce există despre el este adresa de e-mail, nu ar trebui să vedeţi nici un fel de text referitor la adresa poştală, telefon, fax sau note personale.

Crearea mecanismului de ştergere a înregistrărilor

Mecanismul de ştergere a înregistrărilor este aproape identic cu scriptul folosit pentru vizualizarea înregistrărilor. De fapt, puteţi să luaţi primele 44 de linii din listingul 18.3 şi să le copiaţi într-un fişier nou, denumit `delentry.php` şi să efectuaţi următoarele modificări:

- pe liniile 7, 37 şi 43, schimbaţi `"view"` cu `"delete"`;
- pe liniile 24 şi 39, schimbaţi `"View"` cu `"Delete"`.

Începând cu linia 45, restul codului corespunzător scriptului `delentry.php` este afişat în listingul 18.4.

LISTINGUL 18.4. Un script denumit `delentry.php` ce poate fi folosit pentru selectarea şi ştergerea unei înregistrări.

```
45:    //verificam completarea campurilor obligatorii
46:      if ($_POST[sel_id] == "")  {
47:        header("Location: delentry.php");
48:        exit;
49:    }
50:
51:    //lanseaza interogarile
52:    $del_master = "delete from master_name where id = $_POST[sel_id]";
53:    mysql_query($del_master);
54:
55:    $del_address = "delete from address where id = $_POST[sel_id]";
56:    mysql_query($del_address);
57:
58:    $del_tel = "delete from telephone where id = $_POST[sel_id]";
59:    mysql_query($del_tel);
60:
61:    $del_fax = "delete from fax where id = $_POST[sel_id]";
62:    mysql_query($del_fax);
63:
64:    $del_email = "delete from email where id = $_POST[sel_id]";
65:    mysql_query($del_email);
66:
67:    $del_note = "delete from personal_notes where id = $_POST[sel_id]";
68:    mysql_query($del_master);
69:
70:    $display_block = "<h1>Record(s) Deleted</h1>
71:    <P>Would you like to
72:    <a href=\"$_SERVER[PHP_SELF]\">delete another</a>?</p>";
73: }
74: ?>
75: <HTML>
76: <HEAD>
77: <TITLE>My Records</TITLE>
78: </HEAD>
79: <BODY>
80: <?php echo $display_block; ?>
81: </BODY>
82: </HTML>
```

Începând cu linia 45, scriptul caută câmpul obligatoriu, `$_POST[sel_id]`. Dacă această valoare nu există, utilizatorul este trimis înapoi la formularul de selecţie. Pe liniile 52–68, interogările folosite şterg toate informaţiile legate de persoana selectată, din toate tabelele. Liniile 70–72 amplasează un mesaj drăguţ în `$display_block` şi scriptul se termină afişând codul HTML pe ecran. Un rezultat afişat de scriptul de ştergere a înregistrărilor poate fi văzut în figura 18.6.

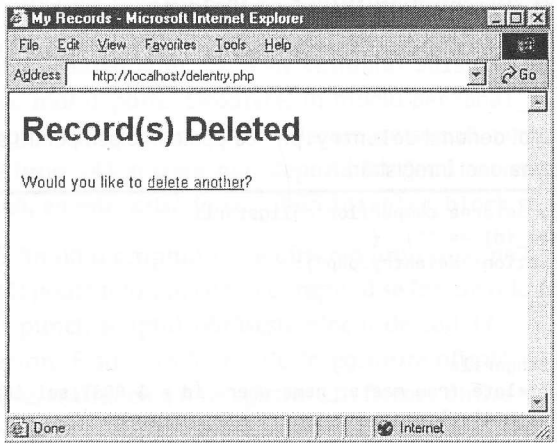


FIGURA 18.6. Ştergerea unei înregistrări.

Acum, haideţi să ne întoarcem la formularul de selecţie şi să observăm că persoana pe care aţi şters-o nu mai există în meniul respectiv.

Adăugarea de înregistrări suplimentare pentru o persoană

Până acum, aţi învăţat să adăugaţi, ştergeţi şi vizualizaţi înregistrări. Ceea ce lipseşte este posibilitatea de lucru cu acele intrări adiţionale din tabelele relaţionate — intrările pentru telefonul de acasă sau de la serviciu, de exemplu. Tot ce trebuie să faceţi sunt câteva modificări la scripturile deja existente.

În scriptul `selentry.php` din listingul 18.3, schimbaţi liniile 153–154 astfel încât acestea să fie:

```
$display_block .= "<P align=center>
<a href=\"addentry.php?master_id=$_POST[sel_id]\">add info</a> ...
<a href=\"$_SERVER[PHP_SELF]\">select another</a></p>";
```

Această schimbare adaugă pur şi simplu un link către scriptul `addentry.php` şi, de asemenea, îi pasează acestuia o variabilă denumită `$master_id`.

Acum trebuie să modificăm scriptul `addentry.php` din listingul 18.2 pentru ca el să facă faţă acestei funcţionalităţi suplimentare. Iată un rezumat al schimbărilor ce trebuie aduse scriptului original.

Înlocuiţi primele 10 linii ale scriptului original `addentry.php` cu fragmentul de cod de mai jos:

```
<?php
if (($_POST[op] != "add") || ($_GET[master_id] != "")) {
    //nu au vazut inca formularul, asa ca il afisam
    $display_block = "
    <h1>Add an Entry</h1>
    <form method=\"post\" action=\"$_SERVER[PHP_SELF]\">";

    if ($_GET[master_id] != "") {
        //conectare la baza de date
        $conn = mysql_connect("localhost", "joeuser", "somepass")
                or die(mysql_error());
        mysql_select_db("testDB",$conn)  or die(mysql_error());

        //preia numele pentru a-l afisa / a-i testa validitatea
        $get_names = "select concat_ws(' ', f_name, l_name) as
                    display_name from master_name where id = $_GET[master_id]";
        $get_names_res = mysql_query($get_names) or die(mysql_error());

        if (mysql_num_rows($get_names_res) == 1) {
            $display_name = mysql_result($get_names_res,Ø,'display_name');
        }
    }

    if ($display_name != "") {
        $display_block .= "<P>Adding information for
                    <strong>$display_name</strong>:</p>";
    } else {
        $display_block .= "
        <P><strong>First/Last Names:</strong><br>
        <input type=\"text\" name=\"f_name\" size=3Ø maxlength=75>
        <input type=\"text\" name=\"l_name\" size=3Ø maxlength=75>";
    }
    $display_block .= "<P><strong>Address:</strong><br>
```

Acest fragment de cod nu face altceva decât să mute elementele formularului, afişând câmpurile corespunzătoare prenumelui şi numelui de familie doar dacă este vorba despre o nouă înregistrare. Dacă scriptul este apelat pentru adăugarea de intrări pentru o înregistrare existentă, numele persoanei este extras din baza de date atât în scop estetic cât şi pentru a se verifica validitatea ID-ului primit.

Apoi, găsiţi această linie:

```
<input type=\"hidden\" name=\"op\" value=\"add\">
```

Sub linie, adăugaţi următoarele:

```
<input type=\"hidden\" name=\"master_id\" value=\"$_GET[master_id]\">
```

Această modificare ne asigură că valoarea cunoscută a lui master_id este pasată mai departe pentru operaţiunea următoare.

Identificaţi care erau liniile 49–67 în scriptul original, care încep cu comentariul time to add tables şi se termină cu obţinerea valorii pentru $master_id. Aceste linii trebuie înlocuite cu următoarele:

```
//momentul sa adaugam tabelele, asa ca verificam campurile obligatorii
if (($_POST[f_name] == "") || ($_POST[l_name] == "")) &&
          ($_POST[master_id] == "")) {
    header("Location: addentry.php");
    exit;
}

//conectare la baza de date
$conn = mysql_connect("localhost", "joeuser", "somepass")
          or die(mysql_error());
mysql_select_db("testDB",$conn)  or die(mysql_error());

if ($_POST[master_id] == "") {
    //adaugare la tabela master_name
    $add_master = "insert into master_name values ('', now(),
                now(), '$_POST[f_name]', '$_POST[l_name]')";
    mysql_query($add_master) or die(mysql_error());
    //se preia master_id pentru a fi folosit la celelalte tabele
    $master_id = mysql_insert_id();
} else {
     $master_id = $_POST[master_id];
}
```

Aceste linii modifică verificarea câmpurilor obligatorii, permiţând scriptului să continue fără valori pentru prenume şi nume, dar numai dacă are o valoare pentru $_POST[master_id]. Apoi, scriptul se conectează la baza de date pentru a efectua toate adăugirile pe care le dorim, dar sare peste adăugarea înregistrării în tabela master_name în cazul în care există o valoare în $_POST[master_id].

La final, în secţiunea de cod care tratează inserarea înregistrării în tabela personal_notes, schimbaţi comanda INSERT cu comanda REPLACE pentru a putea trata o actualizare a câmpului notes.

Noul script ar trebui să arate ca în listingul 18.5

LISTINGUL 18.5. Noul script addentry.php.

```
 1: <?php
 2: if (($_POST[op] != "add") || ($_GET[master_id] != "")) {
 3:    //nu au vazut inca formularul, asa ca il afisam
 4:    $display_block = "
 5:    <h1>Add an Entry</h1>
 6:    <form method=\"post\" action=\"$_SERVER[PHP_SELF]\">";
 7:
 8:     if ($_GET[master_id] != "") {
```

LISTINGUL 18.5. (*continuare*)

```
9:        //conectare la baza de date
10:       $conn = mysql_connect("localhost", "joeuser", "somepass")
11:                or die(mysql_error());
12:       mysql_select_db("testDB",$conn)  or die(mysql_error());
13:
14:       //preia numele pentru a-l afisa / a-i testa validitatea
15:       $get_names = "select concat_ws(' ', f_name, l_name) as
16:              display_name from master_name where id = $_GET[master_id]";
17:       $get_names_res = mysql_query($get_names) or die(mysql_error());
18:
19:       if (mysql_num_rows($get_names_res) == 1) {
20:           $display_name = mysql_result($get_names_res,0,'display_name');
21:       }
22:   }
23:
24:   if ($display_name != "") {
25:       $display_block .= "<P>Adding information for
26:                     <strong>$display_name</strong>:</p>";
27:   } else {
28:       $display_block .= "
29:     <P><strong>First/Last Names:</strong><br>
30:     <input type=\"text\" name=\"f_name\" size=30 maxlength=75>
31:     <input type=\"text\" name=\"l_name\" size=30 maxlength=75>";
32: }
33: $display_block .= "<P><strong>Address:</strong><br>
34:    <input type=\"text\" name=\"address\" size=30>
35:
36:    <P><strong>City/State/Zip:</strong><br>
37:    <input type=\"text\" name=\"city\" size=30 maxlength=50>
38:    <input type=\"text\" name=\"state\" size=5 maxlength=2>
39:    <input type=\"text\" name=\"zipcode\" size=10 maxlength=10>
40:
41:    <P><strong>Address Type:</strong><br>
42:    <input type=\"radio\" name=\"add_type\" value=\"home\" checked> home
43:    <input type=\"radio\" name=\"add_type\" value=\"work\"> work
44:   <input type=\"radio\" name=\"add_type\" value=\"other\"> other
45:
46:   <P><strong>Telephone Number:</strong><br>
47:   <input type=\"text\" name=\"tel_number\" size=30 maxlength=25>
48:   <input type=\"radio\" name=\"tel_type\" value=\"home\" checked> home
49:   <input type=\"radio\" name=\"tel_type\" value=\"work\"> work
50:   <input type=\"radio\" name=\"tel_type\" value=\"other\"> other
51:
52:   <P><strong>Fax Number:</strong><br>
53:   <input type=\"text\" name=\"fax_number\" size=30 maxlength=25>
54:   <input type=\"radio\" name=\"fax_type\" value=\"home\" checked> home
55:   <input type=\"radio\" name=\"fax_type\" value=\"work\"> work
56:   <input type=\"radio\" name=\"fax_type\" value=\"other\"> other
57:
58:   <P><strong>Email Address:</strong><br>
59:   <input type=\"text\" name=\"email\" size=30 maxlength=150>
60:   <input type=\"radio\" name=\"email_type\" value=\"home\" checked> home
61:   <input type=\"radio\" name=\"email_type\" value=\"work\"> work
62:   <input type=\"radio\" name=\"email_type\" value=\"other\"> other
63:
64:   <P><strong>Personal Note:</strong><br>
```

LISTINGUL 18.5. (*continuare*)

```
65:    <textarea name=\"note\" cols=35 rows=5 wrap=virtual></textarea>
66:    <input type=\"hidden\" name=\"op\" value=\"add\">
67:    <input type=\"hidden\" name=\"master_id\" value=\"$_GET[master_id]\">
68:
69:    <p><input type=\"submit\" name=\"submit\" value=\"Add Entry\"></p>
70:    </FORM>";
71:
72: } else if ($_POST[op] == "add") {
73:   //momentul sa adaugam tabelele, asa ca verificam campurile obligatorii
74:   if ((($_POST[f_name] == "") || ($_POST[l_name] == "")) &&
75:   ($_POST[master_id] == "")) {
76:        header("Location: addentry.php");
77:        exit;
78:   }
79:
80:   //conectare la baza de date
81:  $conn = mysql_connect("localhost", "joeuser", "somepass")
82:  or die(mysql_error());
83:  mysql_select_db("testDB",$conn)  or die(mysql_error());
84:
85:   if ($_POST[master_id] == "") {
86:       //adaugare la tabela master_name
87:       $add_master = "insert into master_name values ('', now(),
88:       now(), '$_POST[f_name]', '$_POST[l_name]')";
89:       mysql_query($add_master) or die(mysql_error());
90:      //se preia master_id pentru a fi folosit la celelalte tabele
91:      $master_id = mysql_insert_id();
92:   } else {
93:      $master_id = $_POST[master_id];
94:   }
95:
96:    if (($_POST[address]) || ($_POST[city]) || ($_POST[state]) ||
97:         ($_POST[zipcode])) {
98:        //informatii relevante, care se adauga la tabela address
99:        $add_address = "insert into address values ('', $master_id,
100:            now(), now(), '$_POST[address]', '$_POST[city]',
101:            '$_POST[state]', '$_POST[zipcode]', '$_POST[add_type]')";
102:       mysql_query($add_address) or die(mysql_error());
103:    }
104:
105:    if ($_POST[tel_number]) {
106:        //informatii relevante, care se adauga la tabela telephone
107:        $add_tel = "insert into telephone values ('', $master_id,
108:         now(), now(), '$_POST[tel_number]',  '$_POST[tel_type]')";
109:        mysql_query($add_tel) or die(mysql_error());
110:    }
111:
112:    if ($_POST[fax_number]) {
113:        //informatii relevante, care se adauga la tabela fax
114:        $add_fax = "insert into fax values ('', $master_id, now(),
115:         now(), '$_POST[fax_number]',  '$_POST[fax_type]')";
116:        mysql_query($add_fax) or die(mysql_error());
117:    }
118:
119:    if ($_POST[email]) {
```

LISTINGUL 18.5. (*continuare*)

```
120:        //informatii relevante, care se adauga la tabela email
121:        $add_email = "insert into email values ('', $master_id,
122:             now(), now(), '$_POST[email]',  '$_POST[email_type]')";
123:        mysql_query($add_email) or die(mysql_error());
124:    }
125:
126:    if ($_POST[note]) {
127:        //informatii relevante, care se adauga la tabela notes
128:        $add_note = "replace into personal_notes values ('', $master_id,
129:             now(), now(), '$_POST[note]')";
130:        mysql_query($add_note) or die(mysql_error());
131:    }
132:
133:    $display_block = "<h1>Entry Added</h1>
134:    <P>Your entry has been added.  Would you like to
135:     <a href=\"addentry.php\">add another</a>?</p>";
136: }
137: ?>
138: <HTML>
139: <HEAD>
140: <TITLE>Add an Entry</TITLE>
141: </HEAD>
142: <BODY>
143: <?php echo $display_block; ?>
144: </BODY>
145: </HTML>
```

Puteţi încerca acest script revizuit selectând o înregistrare pentru a o vizualiza şi dând clic, apoi, pe linkul `add info`. Ar trebui să vedeţi un formular ca cel din figura 18.7.

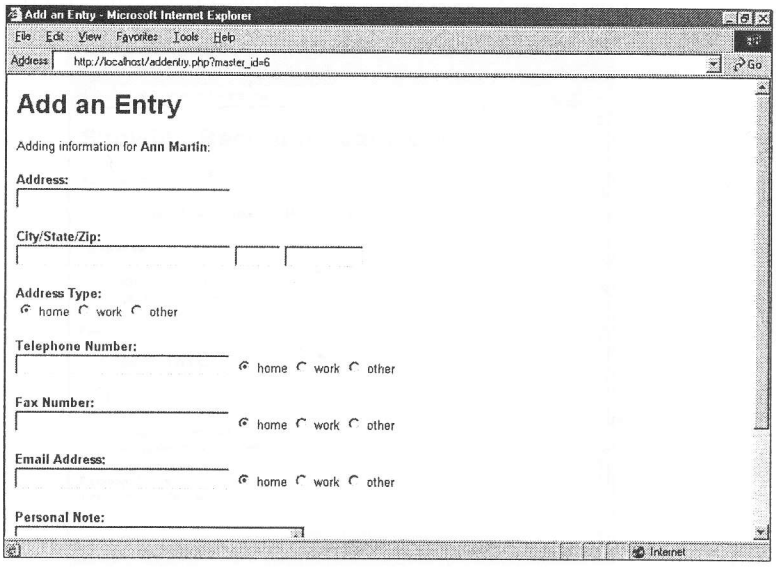


FIGURA 18.7. Adăugarea unor înregistrări.

După ce trimiteţi acest formular, puteţi să vă întoarceţi la secvenţa de selectare şi să vizualizaţi înregistrarea pentru a verifica dacă schimbările dumneavoastră au fost operate.

Rezumat

În cadrul acestui capitol practic, aţi aplicat cunoştinţele dumneavoastră de bază asupra folosirii MySQL şi PHP pentru a crea o agendă de adrese personală. Aţi învăţat cum să creaţi tabelele din baza de date şi scripturile necesare pentru adăugarea, ştergerea şi vizualizarea înregistrărilor. Aţi mai învăţat şi procesul de adăugare a mai multor înregistrări ataşate unei singure înregistrări principale.

Atelier

Atelierul este conceput să anticipeze posibile întrebări, să vă ajute să recapitulaţi cele învăţate şi să începeţi să vă puneţi cunoştinţele în practică.

Test

1. Când pasaţi o variabilă prin intermediului şirului de interogare HTTP, care dintre vectorii superglobali conţine această variabilă?
2. Câte înregistrări puteţi avea în tabelele `address`, `email`, `telephone` şi `fax` pentru fiecare persoană din tabela `master_name`?

Soluţii

1. Vectorul superglobal `$_GET`.
2. Oricât de multe doriţi — este o bază de date relaţională!

Activităţi

1. Modificaţi fiecare dintre scripturile de administrare astfel încât să fie afişat un link către meniul principal în partea de jos a fiecărei pagini.
2. Folosiţi cea de-a doua versiune a scriptului `addentry.php` pentru a adăuga o informaţie de contact suplimentară în baza dumneavoastră de date. În figura 18.8 vedeţi cum trebuie să arate o înregistrare după adăugarea unei informaţii de contact suplimentare.

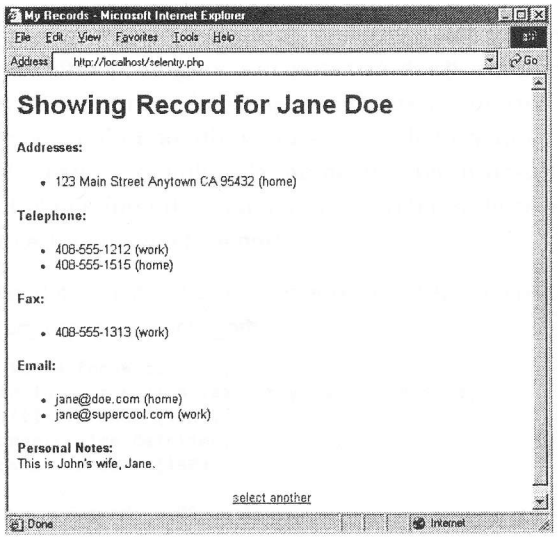
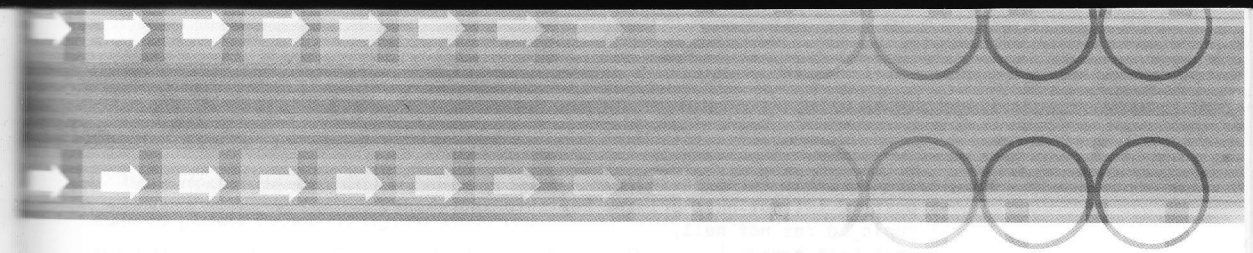


FIGURA 18.8. Înregistrarea corespunzătoare unei persoane cu mai multe intrări în tabele.



CAPITOLUL 19
Crearea unui forum de discuții simplu

În acest capitol, veți învăța procesul de proiectare necesar pentru crearea unui forum de discuții simplu. Acesta include dezvoltarea tabelelor bazei de date, formulare de introducere a datelor pentru utilizator și mecanismul de afișare a rezultatelor. Atunci când este astfel fragmentată, o asemenea sarcină pare simplă — și chiar este!

În acest capitol, veți învăța cum să:

- creați tabele, formulare de introducere a datelor și de afișare a rezultatelor pentru un forum de discuții simplu.

Proiectarea tabelelor bazei de date

Gândiți-vă la componentele de bază ale unui forum: subiectele (`topics`) și mesajele (`posts`). Un forum — dacă este utilizat corespunzător de administratorii săi — trebuie să conțină câteva subiecte, iar fiecare dintre acestea va avea o serie de mesaje publicate de diverși utilizatori. Știind aceste lucruri, trebuie să vă dați seama că mesajele sunt legate de subiecte prin intermediul unui câmp cheie. Această cheie formează relația dintre două tabele.

Gândiți-vă la elementele implicate de subiecte. Aveți, fără îndoială, nevoie de un câmp pentru titlu și, în plus, s-ar putea să doriți câmpuri suplimentare în care să stocați data creării și datele de identificare ale utilizatorului care a creat subiectul. În mod similar, gândiți-vă la elementele necesare pentru mesaje: veți avea nevoie de textul mesajului, de momentul creării acestuia și de persoana care l-a creat. Cel mai important lucru este că aveți nevoie de o cheie care să lege mesajul de subiectul din care face parte.

Următoarele două instrucțiuni de creare a tabelelor creează aceste tabele, denumite `forum_topics` și `forum_posts`:

```
mysql> create table forum_topics (
    -> topic_id int not null primary key auto_increment,
    -> topic_title varchar (150),
    -> topic_create_time datetime,
    -> topic_owner varchar (150)
    -> );
Query OK, 0 rows affected (0.03 sec)
```

```
mysql> create table forum_posts (
    -> post_id int not null primary key auto_increment,
    -> topic_id int not null,
    -> post_text text,
    -> post_create_time datetime,
    -> post_owner varchar (150)
    -> );
Query OK, 0 rows affected (0.00 sec)
```

Apropo

> În acest exemplu de forum, vom identifica utilizatorii pe baza adresei lor de e-mail şi nu va fi necesară nici o procedură de autentificare.

Ar trebui să aveţi acum două tabele vide, care abia aşteaptă să fie populate cu date. În secţiunea următoare veţi crea formularele de introducere a datelor pentru adăugarea unui subiect şi a unui mesaj.

Crearea formularelor şi scripturilor pentru introducerea datelor

Înainte să puteţi adăuga vreun mesaj, trebuie să adăugaţi forumului un subiect. Este o practică uzuală la crearea unui forum ca primul subiect şi primul mesaj să fie adăugate în acelaşi timp. Din punctul de vedere al utilizatorului, nu prea are sens să adauge mai întâi un subiect şi apoi să se întoarcă pentru a adăuga şi primul mesaj. Trebuie să încercaţi să oferiţi un proces cât mai simplu cu putinţă.

Listingul 19.1 arată formularul de creare a unui subiect, care include şi un spaţiu pentru primul mesaj din cadrul respectivului subiect.

LISTINGUL 19.1. Formularul de adăugare a unui subiect.

```
 1: <html>
 2: <head>
 3: <title>Add a Topic</title>
 4: </head>
 5: <body>
 6: <h1>Add a Topic</h1>
 7: <form method=post action="do_addtopic.php">
 8: <p><strong>Your E-Mail Address:</strong><br>
 9: <input type="text" name="topic_owner" size=40 maxlength=150>
10: <p><strong>Topic Title:</strong><br>
11: <input type="text" name="topic_title" size=40 maxlength=150>
12: <P><strong>Post Text:</strong><br>
13: <textarea name="post_text" rows=8 cols=40 wrap=virtual></textarea>
14: <P><input type="submit" name="submit" value="Add Topic"></p>
15: </form>
16: </body>
17: </html>
```

Pare destul de simplu — cele trei câmpuri afişate în cadrul formularului, pe care le puteţi vedea în figura 19.1, reprezintă toate datele de care aveţi nevoie pentru a completa ambele tabele; scriptul şi baza de date pot completa restul. Salvaţi listingul 19.1 într-un fişier cu titlul `addtopic.html` şi puneţi-l în directorul rădăcină al serverului dumneavoastră Web pentru a putea urmări ce vom face în continuare.

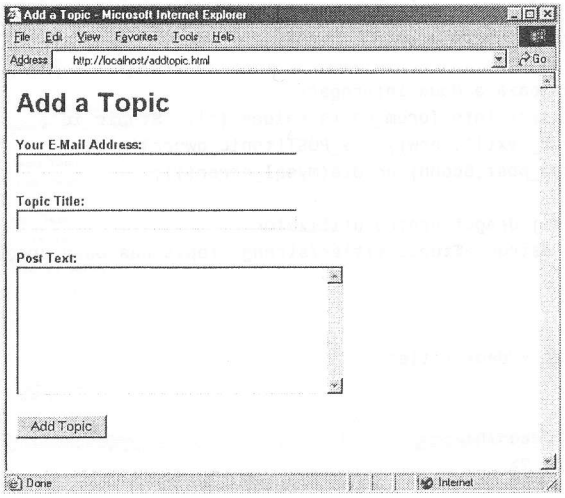


FIGURA 19.1. Formularul pentru crearea subiectului.

Pentru a crea o intrare în tabela `forum_topics`, veţi folosi câmpurile `topic_title` şi `topic_owner` din formularul de introducere a datelor. Câmpurile `topic_id` şi `topic_create_time` vor fi completate automat. La fel, în tabela `forum_posts`, folosiţi câmpurile post_text şi `topic_owner` din formularul transmis de utilizator, iar câmpurile `post_id`, `post_create_time` şi `topic_id` vor fi completate automat. Deoarece aveţi nevoie de o valoare pe care să o introduceţi în câmpul `topic_id` pentru a putea insera înregistrarea în tabela `forum_posts`, înseamnă că interogarea care face acest lucru trebuie să vină după cea care inserează înregistrarea în tabela `forum_topics`.

Listingul 19.2 creează scriptul care adaugă această înregistrare în tabelă.

LISTINGUL 19.2. Scriptul de adăugare a subiectului.

```
 1: <?php
 2: //verificam completarea campurilor obligatorii
 3: if ((!$_POST[topic_owner]) || (!$_POST[topic_title])
 4:     || (!$_POST[post_text])) {
 5:    header("Location: addtopic.html");
 6:    exit;
 7: }
 8:
 9: //se conecteaza la server si selecteaza baza de date
10: $conn = mysql_connect("localhost", "joeuser", "somepass")
```

LISTINGUL 19.2. (*continuare*)

```
11:     or die(mysql_error());
12: mysql_select_db("testDB",$conn)  or die(mysql_error());
13:
14: //creaza si lanseaza prima interogare
15: $add_topic = "insert into forum_topics values ('', '$_POST[topic_title]',
16:     now(), '$_POST[topic_owner]')";
17: mysql_query($add_topic,$conn) or die(mysql_error());
18:
19: //preia id-ul ultimei interogari
20: $topic_id = mysql_insert_id();
21:
22: //creaza si lanseaza a doua interogare
23: $add_post = "insert into forum_posts values ('', '$topic_id',
24:     '$_POST[post_text]', now(), '$_POST[topic_owner]')";
25: mysql_query($add_post,$conn) or die(mysql_error());
26:
27: //creaza un mesaj dragut pentru utilizator
28: $msg = "<P>The <strong>$topic_title</strong> topic has been created.</p>";
29: ?>
30: <html>
31: <head>
32: <title>New Topic Added</title>
33: </head>
34: <body>
35: <h1>New Topic Added</h1>
36: <?php echo $msg; ?>
37: </body>
38: </html>
```

Liniile 3–7 verifică existenţa celor trei câmpuri obligatorii de care avem nevoie pentru a completa ambele tabele. Dacă oricare dintre acestea lipseşte, utilizatorul este trimis înapoi la formularul iniţial.

Liniile 10–20 realizează conexiunea la baza de date, lucru care ar trebui să vă fie cunoscut deja. Liniile 15–17 creează şi inserează prima interogare, ce adaugă subiectul în tabela `forum_topics`. Observaţi că primul câmp este lăsat gol, astfel încât numărul auto-incrementat să fie adăugat de către sistem. În mod similar, funcţia `now()` este folosită pentru a marca înregistrarea cu data şi ora curentă. Celelalte câmpuri ale înregistrării sunt completate folosind valorile din formular.

Pe linia 20 puteţi vedea la lucru o funcţie foarte utilă: `mysql_insert_id()`. Această funcţie extrage ID-ul din cheia primară a ultimei înregistrări inserate de către script în baza de date. În acest caz, `mysql_insert_id()` preia valoarea câmpului id din tabela `forum_topics`, care va deveni valoarea pentru câmpul `topic_id` din tabela `forum_posts`.

Liniile 25–30 creează şi inserează cel de-a doua interogare, folosind, din nou, un amestec de informaţii cunoscute şi furnizate de către sistem. Linia 28 nu face altceva decât să creeze un mesaj pentru utilizator, iar restul scriptului afişează informaţii pe ecran.

Salvaţi acest listing sub numele `do_addtopic.php` — numele acţiunii din scriptul precedent — şi amplasaţi fişierul în directorul rădăcină al serverului dumneavoastră Web. Completaţi formularul, apoi transmiteţi-l şi va trebui să vedeţi mesajul New Topic Added (subiect nou adăugat). Figurile 19.2 şi 19.3 arată desfăşurarea evenimentelor.

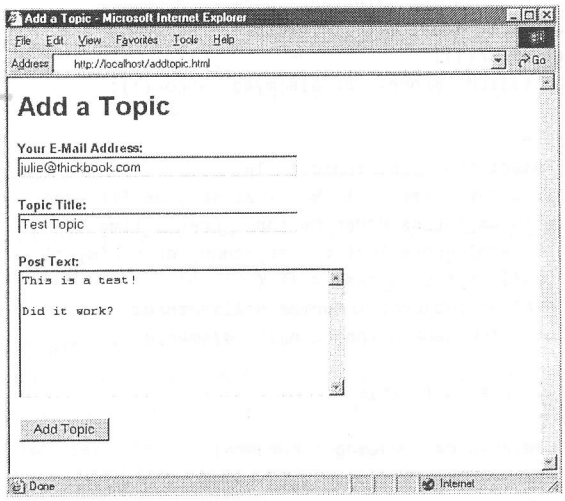


FIGURA 19.2. Adăugarea unui subiect şi a primului mesaj.

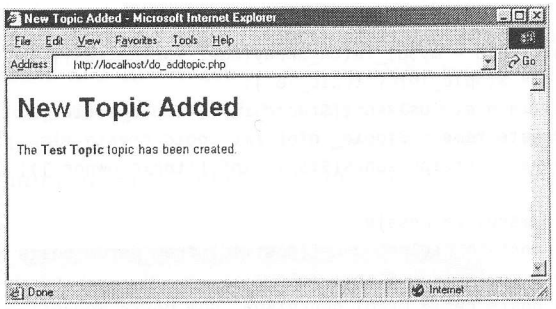


FIGURA 19.3. Adăugarea reuşită a subiectului şi a primului mesaj.

În secţiunea următoare, veţi completa acest puzzle cu încă două noi piese: afişarea subiectelor şi a mesajelor şi introducerea unui răspuns la un mesaj.

Afişarea listei de subiecte

Acum că aveţi un subiect şi cel puţin un mesaj în baza dumneavoastră de date, puteţi afişa informaţiile şi le puteţi da oamenilor posibilitatea de a adăuga noi subiecte sau de a răspunde în cadrul celor existente. În listingul 19.3, facem un pas înapoi şi creăm pagina de listare a subiectelor. Această pagină va afişa informaţii

de bază despre fiecare subiect şi îi va pune utilizatorului la dispoziţie un link pentru adăugarea unui subiect nou; deja aveţi formularul şi scriptul pentru efectuarea acestei acţiuni. Acest script va fi, de fapt, pagina de intrare a forumului dumneavoastră.

LISTINGUL 19.3. Scriptul de afişare a subiectelor.

```
<?php
//se conecteaza la server si selecteaza baza de date
$conn = mysql_connect("localhost", "joeuser", "somepass")
    or die(mysql_error());
mysql_select_db("testDB",$conn)  or die(mysql_error());

//preia subiectele
$get_topics = "select topic_id, topic_title,
date_format(topic_create_time,  '%b %e %Y at %r') as fmt_topic_create_time,
topic_owner from forum_topics order by topic_create_time desc";
$get_topics_res = mysql_query($get_topics,$conn) or die(mysql_error());
if (mysql_num_rows($get_topics_res) < 1) {
   //nu exista nici un subiect; informam utilizatorul
   $display_block = "<P><em>No topics exist.</em></p>";
} else {
   //creaza sirul care va fi afisat
   $display_block = "
   <table cellpadding=3 cellspacing=1 border=1>
   <tr>
   <th>TOPIC TITLE</th>
   <th># of POSTS</th>
   </tr>";

    while ($topic_info = mysql_fetch_array($get_topics_res)) {
       $topic_id = $topic_info['topic_id'];
       $topic_title = stripslashes($topic_info['topic_title']);
       $topic_create_time = $topic_info['fmt_topic_create_time'];
       $topic_owner = stripslashes($topic_info['topic_owner']);

       //preia numarul de mesaje
       $get_num_posts = "select count(post_id) from forum_posts
            where topic_id = $topic_id";
       $get_num_posts_res = mysql_query($get_num_posts,$conn)
            or die(mysql_error());
       $num_posts = mysql_result($get_num_posts_res,0,'count(post_id)');

       //se adauga la textul de afisat
       $display_block .= "
       <tr>
       <td><a href=\"showtopic.php?topic_id=$topic_id\">
            <strong>$topic_title</strong></a><br>
       Created on $topic_create_time by $topic_owner</td>
       <td align=center>$num_posts</td>
       </tr>";
   }

   //inchide tabela
```

LISTINGUL 19.3. (*continuare*)

```
48:     $display_block .= "</table>";
49: }
50: ?>
51: <html>
52: <head>
53: <title>Topics in My Forum</title>
54: </head>
55: <body>
56: <h1>Topics in My Forum</h1>
57: <?php echo $display_block; ?>
58: <P>Would you like to <a href="addtopic.html">add a topic</a>?</p>
59: </body>
60: </html>
```

Deşi listingul 19.3 pare să conţină mult cod, el constă, de fapt, dintr-o serie de concepte simple pe care le-aţi întâlnit deja. Liniile 3–5 realizează conexiunea la baza de date, pregătind transmiterea de interogări. Liniile 8–10 reprezintă primele astfel de interogări, ele selectând toate informaţiile despre subiecte, ordonate descrescător după dată. Cu alte cuvinte, subiectul care a fost creat ultimul (cel mai nou dintre subiecte) va fi afişat în capul listei. În cadrul interogării, observaţi folosirea funcţiei `date_format()` pentru afişarea datei într-un format mult mai atractiv decât cel stocat în baza de date.

Linia 12 verifică dacă interogarea a returnat vreo înregistrare. Dacă nu a rezultat nici o interogare, adică nu există nici un subiect în tabelă, veţi dori să îi spuneţi utilizatorului acest lucru. Linia 14 creează acest mesaj. În acest moment, dacă nu ar exista nici un subiect, scriptul ar părăsi blocul de cod `if...else` şi-ar încheia execuţia; următoarea acţiune s-ar petrece pe linia 51, unde începe codul HTML static. Dacă scriptul s-ar opri aici, mesajul creat pe linia 14 ar fi afişat pe linia 57, iar dumneavoastră aţi vedea ceva asemănător cu figura 19.4.

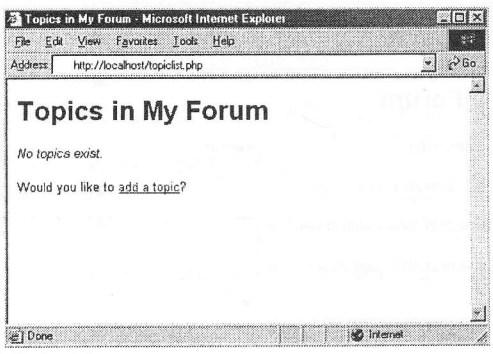


FIGURA 19.4. Nu a fost găsit nici un subiect.

Dacă, pe de altă parte, aveţi subiecte în tabela dumneavoastră `forum_topics`, scriptul continuă cu linia 15. Pe linia 17, este început un bloc de text care conţine începutul unui tabel HTML.

Liniile 18–22 iniţializează tabelul cu două coloane: una pentru titlu şi una pentru numărul de mesaje. Blocul de text este lăsat în pace momentan, iar, pe linia 24, începem să parcurgem iterativ rezultatele interogării iniţiale.

Bucla `while` de pe linia 24 spune că atâta vreme cât există elemente de extras din setul de rezultate, fiecare rând trebuie extras într-un vector denumit `$topic_info` şi numele câmpurilor trebuie folosite drept elemente pentru acest vector în vedere atribuirii valorilor lor unor variabile noi. Astfel, primul element pe care încercăm să îl extragem este câmpul `topic_id`, pe linia 25. Atribuim variabilei `$topic_id` valoarea elementului `$topic_info['topic_id']`, ceea ce înseamnă că obţinem o variabilă locală dintr-un vector denumit `$topic_info` ce conţine un element cu numele `topic_id`. Continuăm să facem acest lucru pentru variabilele `$topic_title`, `$topic_create_time` şi `$topic_owner` pe liniile 26–28. Funcţia `stripslashes()` elimină orice caractere de eludare care ar putea să existe în tabelă.

Pe liniile 31–35, emitem o nouă interogare, în contextul buclei `while`, pentru a obţine numărul de mesaje din cadrul subiectului respectiv. Pe linia 38 continuăm construirea şirului de caractere `$display_block`, folosind operatorul de concatenare (`.=`) pentru a ne asigura că adăugirile sunt dispuse la sfârşitul şirului de caractere creat până în acel moment. Cea de-a doua coloană a tabelului HTML, de pe linia 43, arată numărul de mesaje. Pe linia 45, părăsim bucla `while`, iar pe linia 48, adăugăm o ultimă bucăţică şirului de caractere `$display_block` pentru a închide tabelul HTML. Restul de linii afişează codul HTML al paginii, inclusiv valoarea şirului de caractere `$display_block`.

Dacă salvaţi acest fişier sub numele `topiclist.php` şi îl în amplasaţi în directorul rădăcină al serverului dumneavoastră Web, dar şi dacă aveţi subiecte în tabelele bazei dumneavoastră de date, veţi vedea ceva asemănător cu figura 19.5.

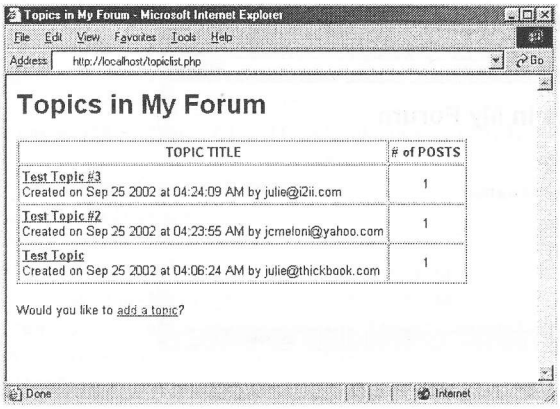


FIGURA 19.5. Subiectele care sunt disponibile.

Afişarea mesajelor din cadrul unui subiect

După cum probabil ați ghicit, următoarea etapă este construirea fișierului showtopic.php, pentru a afișa mesajele din cadrul unui subiect. Listingul 19.4 face exact acest lucru.

LISTINGUL 19.4. Un script care afişează mesajele din cadrul unui subiect.

```
<?php
//verifica informatiile din sirul de interogare
if (!$_GET[topic_id]) {
    header("Location: topiclist.php");
    exit;
}

//se conecteaza la server si selecteaza baza de date
$conn = mysql_connect("localhost", "joeuser", "somepass")
    or die(mysql_error());
mysql_select_db("testDB",$conn)  or die(mysql_error());

//verifica daca subiectul exista
$verify_topic = "select topic_title from forum_topics where
    topic_id = $_GET[topic_id]";
$verify_topic_res = mysql_query($verify_topic, $conn)
    or die(mysql_error());

if (mysql_num_rows($verify_topic_res) < 1) {
    //acest subiect nu exista
    $display_block = "<P><em>You have selected an invalid topic.
    Please <a href=\"topiclist.php\">try again</a>.</em></p>";
} else {
    //preia titlul subiectului
    $topic_title = stripslashes(mysql_result($verify_topic_res,0,
        'topic_title'));

    //identifica mesajele corespunzatoare subiectului
    $get_posts = "select post_id, post_text, date_format(post_create_time,
        '%b %e %Y at %r') as fmt_post_create_time, post_owner from
        forum_posts where topic_id = $_GET[topic_id]
        order by post_create_time asc";

    $get_posts_res = mysql_query($get_posts,$conn) or die(mysql_error());

    //creaza sirul care va fi afisat
    $display_block = "
    <P>Showing posts for the <strong>$topic_title</strong> topic:</p>

    <table width=100% cellpadding=3 cellspacing=1 border=1>
    <tr>
    <th>AUTHOR</th>
    <th>POST</th>
    </tr>";

    while ($posts_info = mysql_fetch_array($get_posts_res)) {
        $post_id = $posts_info['post_id'];
```

LISTINGUL 19.4. *(continuare)*

```
48:         $post_text = nl2br(stripslashes($posts_info['post_text']));
49:         $post_create_time = $posts_info['fmt_post_create_time'];
50:         $post_owner = stripslashes($posts_info['post_owner']);
51:
52:         //afisare
53:         $display_block .= "
54:         <tr>
55:         <td width=35% valign=top>$post_owner<br>[$post_create_time]</td>
56:         <td width=65% valign=top>$post_text<br><br>
57:          <a href=\"replytopost.php?post_id=$post_id\"><strong>REPLY TO
58:          POST</strong></a></td>
59:         </tr>";
60:     }
61:
62:     //inchide tabela
63:     $display_block .= "</table>";
64: }
65: ?>
66: <html>
67: <head>
68: <title>Posts in Topic</title>
69: </head>
70: <body>
71: <h1>Posts in Topic</h1>
72: <?php echo $display_block; ?>
73: </body>
74: </html>
```

În listingul 19.4, liniile 3–6 verifică dacă există o valoare pentru `topic_id` în şirul de interogare `HTTP` primit prin metoda `GET`. Deoarece afişăm toate mesajele din subiectul selectat, trebuie să ştim despre ce subiect este vorba, iar aceasta este modalitatea în care ne este transmisă informaţia. Dacă nu există o valoare în elementul `$_GET[topic_id]`, utilizatorul este trimis înapoi către pagina de afişare a subiectelor.

Dacă am trecut cu succes de verificarea existenţei lui `topic_id`, liniile 9–11 realizează conexiunea la baza de date, pregătind emiterea de interogări. Liniile 14–17 conţin prima interogare SQL, iar aceasta este folosită pentru a verifica validitatea valorii `topic_id` ce este primită prin metoda `GET`, prin selectarea câmpului `topic_title` corespunzător respectivului subiect. Dacă verificarea validităţii nu reuşeşte pe linia 19, pe liniile 21–22 este creat un mesaj, iar scriptul părăseşte blocul de cod `if...else` şi îşi încheie execuţia afişând codul HTML. Rezultatul, în acest caz, arată ca în figura 19.6.

Dacă, pe de altă parte, subiectul este valid, extragem, pe linia 25, valoarea câmpului `topic_title`, folosind din nou funcţia `stripslashes()` pentru a elimina orice caractere de eludare. Apoi, pe liniile 29–32, este emisă o interogare SQL care extrage toate mesajele asociate cu subiectul în chestiune, ordonate crescător, după data şi ora creării. În acest caz, mesajele cele mai noi se află în partea de jos a listei. Pe linia 37, este început un bloc de text ce conţine începutul unui tabel HTML. Liniile 40–44 iniţializează tabelul cu două coloane: una pentru autorul mesajului şi alta pentru textul efectiv al mesajului.

Blocul de text este abandonat momentan, iar, pe linia 46, începem să parcurgem iterativ rezultatele interogării inițiale.

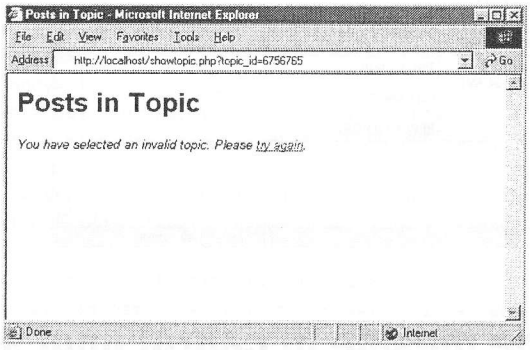


FIGURA 19.6. A fost selectat un subiect invalid.

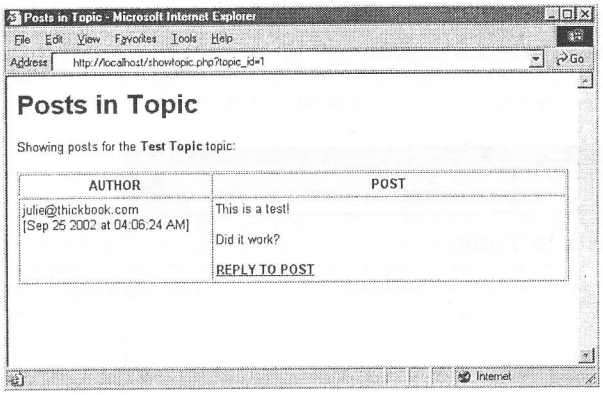
Bucla `while` care începe pe linia 46 spune că atâta vreme cât există elemente de extras din setul de rezultate, trebuie să fie extras fiecare rând sub forma unui vector denumit `$posts_info` și numele câmpurilor trebuie folosite ca elemente ale vectorului în vederea extragerii valorilor acestora în variabile noi. Astfel, primul element pe care încercăm să îl extragem este câmpul `post_id`, pe linia 47. Atribuim variabilei `$post_id` valoarea elementului `$posts_info['post_id']`, ceea ce înseamnă că obținem o variabilă locală dintr-un vector denumit `$posts_info` ce conține un element cu numele `post_id`. Continuăm să facem acest lucru pentru variabilele `$post_text`, `$post_create_time` și `$post_owner` pe liniile 48–50. Funcția `stripslashes()` este folosită din nou pentru a elimina orice caractere de eludare, iar funcția `nl2br()` este folosită asupra valorii elementului `$posts_info['post_text']` pentru a înlocui toate caracterele de linie nouă cu marcaje HTML `<br>`.

Pe linia 53, continuăm construirea șirului de caractere `$display_block`, folosind operatorul de concatenare (.=) pentru a ne asigura că adăugirile sunt dispuse la sfârșitul șirului de caractere creat până în acel moment. Pe linia 54, creăm coloana tabelului HTML pentru afișarea autorului și a datei creării mesajului. Cea de-a doua coloană a tabelului HTML, pe linia 56, afișează textul mesajului precum și un link ce poate fi folosit pentru a răspunde la mesaj. Pe linia 60, părăsim bucla `while`, iar, pe linia 63, adăugăm o ultimă bucățică șirului `$display_block` pentru a închide tabelul HTML. Restul liniilor afișează codul HTML al paginii, inclusiv valoarea șirului de caractere `$display_block`.

Dacă salvați acest fișier sub numele `showtopic.php` și în amplasați în directorul rădăcină al serverului dumneavoastră Web și dacă aveți mesaje în tabelele bazei dumneavoastră de date, veți vedea ceva asemănător cu figura 19.7.

Un subiect care conține un singur mesaj nu este interesant deloc, așa că haideți să încheiem acest capitol creând un script care să permită adăugarea unui mesaj în cadrul unui subiect.

FIGURA 19.7. Mesajele din cadrul unui subiect.



Adăugarea de mesaje în cadrul unui subiect

În această etapă finală, veţi crea scriptul `replytopost.php`, care arată extrem de similar cu formularul şi scriptul folosite pentru adăugarea unui subiect. Listingul 19.5 arată codul pentru acest fişier combinat — script şi formular.

LISTINGUL 19.5. Scriptul pentru adăugarea de răspunsuri în cadrul unui subiect.

```
<?php
//se conecteaza la server si selecteaza baza de date; vom avea nevoie curand
$conn = mysql_connect("localhost", "joeuser", "somepass")
    or die(mysql_error());
mysql_select_db("testDB",$conn)  or die(mysql_error());

//verifica daca trebuie aratat formularul, sau daca se adauga un mesaj
if ($_POST[op] != "addpost") {
   // aratam formularul; se verifica daca sirul de interogare este complet
   if (!$_GET[post_id]) {
      header("Location: topiclist.php");
      exit;
   }

     //trebuie sa verificam existenta subiectului si a mesajului
   $verify = "select ft.topic_id, ft.topic_title from
    forum_posts as fp left join forum_topics as ft on
    fp.topic_id = ft.topic_id where fp.post_id = $_GET[post_id]";

   $verify_res = mysql_query($verify, $conn) or die(mysql_error());
    if (mysql_num_rows($verify_res) < 1) {
       //acest mesaj sau subiect nu exista
       header("Location: topiclist.php");
       exit;
   } else {
       //preia id-ul si titlul subiectului
       $topic_id = mysql_result($verify_res,0,'topic_id');
       $topic_title = stripslashes(mysql_result($verify_res,
        0,'topic_title'));
```

LISTINGUL 19.5. *(continuare)*

```
30:
31:       echo "
32:       <html>
33:       <head>
34:       <title>Post Your Reply in $topic_title</title>
35:       </head>
36:       <body>
37:       <h1>Post Your Reply in $topic_title</h1>
38:       <form method=post action=\"$_SERVER[PHP_SELF]\">
39:
40:       <p><strong>Your E-Mail Address:</strong><br>
41:       <input type=\"text\" name=\"post_owner\" size=40 maxlength=150>
42:
43:       <P><strong>Post Text:</strong><br>
44:       <textarea name=\"post_text\" rows=8 cols=40 wrap=virtual></textarea>
45:
46:       <input type=\"hidden\" name=\"op\" value=\"addpost\">
47:       <input type=\"hidden\" name=\"topic_id\" value=\"$topic_id\">
48:
49:       <P><input type=\"submit\" name=\"submit\" value=\"Add Post\"></p>
50:
51:       </form>
52:       </body>
53:       </html>";
54:    }
55: } else if ($_POST[op] == "addpost") {
56:    //verifica prezenta elementelor obligatorii in formular
57:    if ((!$_POST[topic_id]) || (!$_POST[post_text]) ||
58:     (!$_POST[post_owner])) {
59:       header("Location: topiclist.php");
60:       exit;
61:    }
62:
63:    //adauga mesajul
64:    $add_post = "insert into forum_posts values ('', '$_POST[topic_id]',
65:     '$_POST[post_text]', now(), '$_POST[post_owner]')";
66:    mysql_query($add_post,$conn) or die(mysql_error());
67:
68:    //redirectioneaza utilizatorul la subiectul respectiv
69:    header("Location: showtopic.php?topic_id=$topic_id");
70:    exit;
71: }
72: ?>
```

Liniile 3–5 realizează conexiunea la baza de date de la începutul scriptului. Deşi scriptul poate să execute operaţiuni diferite în funcţie de starea formularului (dacă este afişat sau transmis), ambele condiţii implică, la un moment dat, interacţiunea cu baza de date.

Linia 8 verifică dacă formularul este transmis. Dacă valoarea lui `$_POST[op]` nu este `"addpost"`, atunci formularul nu a fost încă transmis. De aceea, el trebuie afişat. Înainte de a afişa formularul, totuşi, trebuie să verificăm existenţa acelui element obligatoriu; liniile 10–13 verifică existenţa unei valori corespunzătoare lui `post_id` în şirul de interogare `GET`. Dacă nu există o valoare în `$_GET[post_id]`, utilizatorul este trimis către pagina de afişare a subiectelor.

Dacă am trecut de verificarea valorii lui `post_id`, liniile 17–20 emit o interogare complicată cu ajutorul căreia sunt extrase valorile corespunzătoare câmpurilor `topic_id` şi `topic_title` din tabela `forum_topics`, pe baza singurei valori cunoscute: aceea a lui `post_id`. Această interogare, pe lângă faptul că validează existenţa mesajului, extrage informaţii de care veţi avea nevoie mai încolo în cadrul scriptului. Liniile 21–24 acţionează în cazul în care rezultatul testului de validitate este negativ, trimiţând utilizatorul, din nou, la pagina `topiclist.php`.

Dacă mesajul este valid, atunci extragem valorile câmpurilor `topic_id` şi `topic_title` pe liniile 27–29, folosind şi de această dată funcţia `stripslashes()` pentru a elimina orice caractere de eludare. După aceasta, este afişat în întregime formularul de adăugare a unui mesaj şi asta este tot ce face scriptul până în momentul în care utilizatorul dă clic pe butonul de transmitere a formularului. Cât despre formular, puteţi observa, pe linia 38, că acţiunea acestuia este `$_SERVER[PHP_SELF]`, ceea ce înseamnă că scriptul nostru va fi apelat din nou atunci când este trimis formularul. Mai există şi două câmpuri ascunse, pe liniile 46 şi 47, în care sunt stocate informaţiile ce sunt necesare noii iteraţii a scriptului.

Mergând mai departe, codul care începe pe linia 55 este executat atunci când scriptul este reîncărcat, iar valoarea lui `$_POST[op]` (unul dintre câmpurile ascunse) este `"addpost"`. Acest bloc de cod verifică prezenţa tuturor câmpurilor obligatorii (pe liniile 57–61) şi, dacă toate sunt prezente, emite o interogare prin intermediul căreia noul mesaj este adăugat în baza de date (pe liniile 64–66), afişând apoi, pentru utilizator noul mesaj inserat (pe liniile 69–70).

Dacă salvaţi acest script sub numele `replytopost.php`, amplasaţi fişierul în directorul rădăcină al serverului dumneavoastră Web şi îl încercaţi, veţi vedea ceva asemănător cu figurile 19.8 şi 19.9.

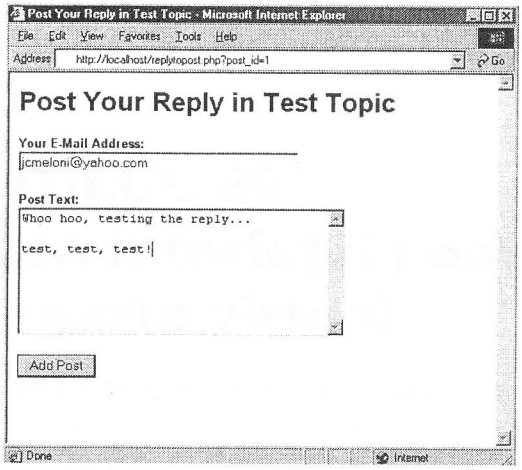


FIGURA 19.8. Ne pregătim să adăugăm un mesaj.

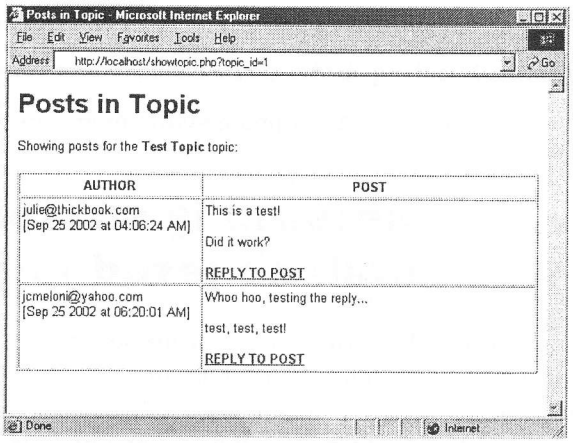


FIGURA 19.9. Un nou mesaj a fost adăugat în listă.

Rezumat

Pentru a parcurge calea de la idee la produsul finit, trebuie să aplicaţi un proces de proiectare. Acest proces înseamnă, în esenţă „gândeşte înainte de a acţiona". Discutaţi regulile, cerinţele şi obiectivele; apoi creaţi versiunea finală a tabelelor dumneavoastră normalizate. În capitolul ce tocmai s-a încheiat, aţi aplicat aceste cunoştinţe pentru crearea unui forum de discuţii simplu, folosind PHP şi MySQL ca să construiţi formulare şi să afişaţi pagini cu subiectele şi mesajele acestui forum.

CAPITOLUL 20
Crearea interfeței pentru un magazin virtual

În cadrul acestei lecții practice, proiectul nostru este crearea unei interfețe generice pentru un magazin virtual. Veți învăța metodele necesare creării tabelelor bazei de date, precum și scripturilor menite să afișeze informații pentru utilizator. Exemplele folosite în acest capitol reprezintă o cale de îndeplinire a acestei sarcini, dar numărul alternativelor este infinit. Scopul acestor exemple este să vă ofere un punct de pornire și nu o metodă finală pentru rezolvarea acestei probleme.

În acest capitol, veți învăța cum să creați:

- tabele relaționale pentru un magazin virtual;
- scripturi care să afișeze categoriile de produse din magazin;
- scripturi care să afișeze produsele individuale.

Planificarea și crearea tabelelor bazei de date

Înainte de aborda procesul de creare a bazei de date pentru un magazin virtual, gândiți-vă la modul în care funcționează un magazin din viața reală. Atunci când intrați în magazin, produsele sunt ordonate într-un fel sau altul: uneltele de grădinărit nu sunt amestecate cu hainele pentru bebeluși, aparatura electronică și detergenții de rufe nu împart același raft și așa mai departe. Aplicând aceste cunoștințe pentru normalizarea bazei de date, vă dați deja seama că aveți nevoie de o tabelă care să conțină categoriile și una care să conțină produsele.

În continuare, gândiți-vă la produse în sine. În funcție de tipul de magazin pe care îl aveți, produsele pot sau nu să aibă diverse culori sau dimensiuni. Dar toate produsele vor avea un nume, o descriere și un preț. Din nou, gândind din punctul de vedere al normalizării, vă dați seama că trebuie să aveți o tabelă generală cu produse și două tabele suplimentare care să fie legate de tabela principală.

Tabelul 20.1 arată o tabelă ilustrativă cu numele câmpurilor pe care o puteți folosi pentru magazinul dumneavoastră virtual. În câteva momente, veți crea instrucțiunile SQL corespunzătoare, dar, mai întâi, uitați-vă la aceste informații și

încercaţi să vă imaginaţi relaţiile dintre ele. Întrebaţi-vă care dintre câmpuri ar trebui să fie chei unice sau primare.

TABELUL 20.1. Tabela şi numele câmpurilor pentru magazinul virtual.

Numele tabelei	Numele câmpurilor
store_categories	id, cat_title, cat_desc
store_items	id, cat_id, item_title, item_price, item_desc, item_image
store_item_size	item_id, item_size
store_item_color	item_id, item_color

După cum puteţi vedea din următoarele instrucţiuni SQL, tabela `store_categories` mai are două câmpuri în afară de câmpul `id`: `cat_title` şi `cat_desc`, pentru titlul categoriei şi, respectiv, pentru descrierea acesteia. Câmpul `id` este cheia primară, iar `cat_title` este un câmp unic deoarece nu există nici un motiv pentru care să aveţi două categorii denumite identic.

```
mysql> create table store_categories (
    -> id int not null primary key auto_increment,
    -> cat_title varchar (50) unique,
    -> cat_desc text
    -> );
Query OK, 0 rows affected (0.03 sec)
```

Tabela `store_items` are cinci câmpuri în afară de câmpul `id`, dintre care nici unul nu reprezintă o cheie unică. Lungimea specificată în definiţia câmpurilor este arbitrară; ar trebui să folosiţi lungimea care este cea mai potrivită pentru magazinul dumneavoastră. Câmpul `cat_id` leagă produsul de o anumită categorie din tabela `store_categories`. Acest câmp nu este unic deoarece veţi dori ca mai multe produse să facă parte dintr-o aceiaşi categorie. Este evident ce date vor conţine câmpurile `item_title`, `item_price` şi `item_desc` (titlul, preţul şi descrierea produsului). Câmpul `item_image` va conţine un nume de fişier — în acest caz se presupune că acest fişier este local, pe calculatorul dumneavoastră — care va fi folosit pentru a construi un marcaj HTML `<IMG>` atunci când trebuie să afişăm informaţiile referitoare la produs.

```
mysql> create table store_items (
    -> id int  not null primary key auto_increment,
    -> cat_id int not null,
    -> item_title varchar (75),
    -> item_price float (8,2),
    -> item_desc text,
    -> item_image varchar (50)
    -> );
Query OK, 0 rows affected (0.00 sec)
```

Atât tabela store_items cât şi tabela store_item_color conţin informaţii opţionale: dacă vindeţi cărţi, dimensiunile sau culorile nu sunt relevante, dar dacă vindeţi tricouri, aceste informaţii sunt esenţiale. Pentru crearea acestor tabele nu este nevoie de nici o cheie, deoarece puteţi asocia oricâte culori şi dimensiuni cu un anumit produs.

```
mysql> create table store_item_size (
    -> item_id int not null,
    -> item_size varchar (25)
    -> );
Query OK, 0 rows affected (0.00 sec)

mysql> create table store_item_color (
    -> item_id int not null,
    -> item_color varchar (25)
    -> );
Query OK, 0 rows affected (0.00 sec)
```

Acestea sunt toate tabelele necesare pentru o interfaţă simplă a unui magazin virtual — adică pentru afişarea produselor pe care le aveţi de vânzare. Capitolul 21, intitulat *Crearea unui mecanism pentru un coş de cumpărături virtual*, introduce şi componenta de interacţiune cu utilizatorul. Deocamdată, concentraţi-vă în special asupra stocurilor de produse.

În capitolul 18, intitulat *Crearea unei agende de adrese online*, aţi învăţat cum să utilizaţi formulare HTML şi scripturi PHP pentru adăugarea şi ştergerea de înregistrări din baza dumneavoastră de date. Dacă aplicaţi aceleaşi principii acestui set de tabele, puteţi să creaţi cu uşurinţă o interfaţă de administrare pentru magazinul dumneavoastră virtual. Nu o să parcurgem acest proces în această carte, dar puteţi să faceţi acest lucru singuri. În acest moment, am toată încrederea că aveţi suficiente cunoştinţe despre PHP şi MySQL astfel încât să puteţi să duceţi la bun sfârşit o astfel de sarcină.

Deocamdată, scrieţi interogări MySQL direct, prin intermediul monitorului MySQL sau cu ajutorul altei interfeţe pentru a adăuga informaţii în tabelele dumneavoastră. Mai jos se găsesc câteva exemple, în cazul în care vreţi să urmăriţi lecţiile folosind aceleaşi date ca cele din exemple.

Inserarea de înregistrări în tabela store_categories

Următoarele interogări creează trei categorii în tabela dumneavoastră store_categories: hats (pălării), shirts (tricouri) şi books (cărţi).

```
mysql> insert into store_categories values
    -> ('1', 'Hats', 'Funky hats in all shapes and sizes!');
Query OK, 1 row affected (0.01 sec)

mysql> insert into store_categories values ('2', 'Shirts', 'From
t-shirts to sweatshirts to polo shirts and beyond, we have them all.');
Query OK, 1 row affected (0.00 sec)
```

```
mysql> insert into store_categories values ('3', 'Books', 'Paperback,
hardback, books for school and books for play, you name it, we have it.');
Query OK, 1 row affected (0.00 sec)
```

În secţiunea următoare, vom adăuga câteva produse în aceste categorii.

Inserarea de înregistrări în tabela `store_items`

Următoarele interogări adaugă înregistrări pentru câte trei produse în fiecare categorie. Dacă doriţi, dumneavoastră puteţi adăuga mai multe produse.

```
mysql> insert into store_items values ('1', '1', 'Baseball Hat', '12.00',
'Fancy, low-profile baseball hat.', 'baseballhat.gif');
Query OK, 1 row affected (0.00 sec)

mysql> insert into store_items values ('2', '1', 'Cowboy Hat', '52.00',
'10 gallon variety', 'cowboyhat.gif');
Query OK, 1 row affected (0.01 sec)

mysql> insert into store_items values ('3', '1', 'Top Hat', '102.00',
'Good for costumes.', 'tophat.gif');
Query OK, 1 row affected (0.00 sec)

mysql> insert into store_items values ('4', '2', 'Short-Sleeved T-Shirt',
'12.00', '100% cotton, pre-shrunk.', 'sstshirt.gif');
Query OK, 1 row affected (0.00 sec)

mysql> insert into store_items values ('5', '2', 'Long-Sleeved T-Shirt',
'15.00', 'Just like the short-sleeved shirt, with longer sleeves.',
'lstshirt.gif');
Query OK, 1 row affected (0.00 sec)

mysql> insert into store_items values ('6', '2', 'Sweatshirt', '22.00',
'Heavy and warm.', 'sweatshirt.gif');
Query OK, 1 row affected (0.00 sec)

mysql> insert into store_items values ('7', '3', 'Jane\'s Self-Help Book',
'12.00', 'Jane gives advice.', 'selfhelpbook.gif');
Query OK, 1 row affected (0.00 sec)

mysql> insert into store_items values ('8', '3', 'Generic Academic Book',
'35.00', 'Some required reading for school, will put you to sleep.',
'boringbook.gif');
Query OK, 1 row affected (0.00 sec)

mysql> insert into store_items values ('9', '3', 'Chicago Manual of Style',
'9.99', 'Good for copywriters.', 'chicagostyle.gif');
Query OK, 1 row affected (0.00 sec)
```

Inserarea de înregistrări în tabela `store_item_size`

Următoarele interogări asociază dimensiuni cu unul dintre cele trei produse din categoria `shirts` din tabela `store_categories`, precum și o dimensiune generică „pentru toate măsurile" cu fiecare dintre produsele din categoria `hats` (probabil că este vorba despre niște pălării mai ciudate). Puteți, apoi, să inserați singuri același set de asocieri cu diverse dimensiuni pentru restul de produse din categoria `shirts`.

```
mysql> insert into store_item_size values (1, 'One Size Fits All');
Query OK, 1 row affected (0.00 sec)

mysql> insert into store_item_size values (2, 'One Size Fits All');
Query OK, 1 row affected (0.00 sec)

mysql> insert into store_item_size values (3, 'One Size Fits All');
Query OK, 1 row affected (0.00 sec)

mysql> insert into store_item_size values (4, 'S');
Query OK, 1 row affected (0.00 sec)

mysql> insert into store_item_size values (4, 'M');
Query OK, 1 row affected (0.00 sec)

mysql> insert into store_item_size values (4, 'L');
Query OK, 1 row affected (0.00 sec)

mysql> insert into store_item_size values (4, 'XL');
Query OK, 1 row affected (0.00 sec)
```

Inserarea de înregistrări în tabela `store_item_color`

Următoarele interogări asociază culori cu unul dintre cele trei produse din categoria `shirts`. Puteți, apoi, să inserați singuri înregistrări privitoare la culori pentru restul de produse din categoriile `shirts` și `hats`.

```
mysql> insert into store_item_color values (1, 'red');
Query OK, 1 row affected (0.00 sec)

mysql> insert into store_item_color values (1, 'black');
Query OK, 1 row affected (0.00 sec)

mysql> insert into store_item_color values (1, 'blue');
Query OK, 1 row affected (0.00 sec)
```

Afișarea categoriilor de produse

Poate că nu vă vine să credeți, dar cea mai complicată etapă a acestui proiect este acum încheiată. Față de proiectarea bazei de date, cu categoriile și produsele ei, crearea scripturilor care să afișeze informațiile este o sarcină ușoară! Primul script pe care îl veți face este acela care enumeră categoriile și produsele. Evident, nu veți dori să afișați vreodată toate categoriile și toate produsele din fiecare categorie în același timp în momentul în care un cumpărător intră pe ușă, ci veți dori să îi dați utilizatorului posibilitatea să aleagă imediat o categorie, să vadă

produsele din cadrul acesteia şi să poată, apoi, să selecteze o altă categorie. Cu alte cuvinte, scriptul acesta va avea două scopuri în acelaşi timp: să afişeze categoriile şi, apoi, să afişeze produsele din cadrul respectivei categorii.

Listingul 20.1 conţine codul scriptului `seestore.php`.

LISTINGUL 20.1. Scriptul pentru vizualizarea categoriilor.

```
<?php
//conectare la baza de date
$conn = mysql_connect("localhost", "joeuser", "somepass")
    or die(mysql_error());
mysql_select_db("testDB",$conn)  or die(mysql_error());

$display_block = "<h1>My Categories</h1>
<P>Select a category to see its items.</p>";

//prezinta pentru inceput categoriile
$get_cats = "select id, cat_title, cat_desc from
    store_categories order by cat_title";
$get_cats_res = mysql_query($get_cats) or die(mysql_error());

if (mysql_num_rows($get_cats_res) < 1) {
   $display_block = "<P><em>Sorry, no categories to browse.</em></p>";
} else {

   while ($cats = mysql_fetch_array($get_cats_res)) {
       $cat_id  = $cats[id];
       $cat_title = strtoupper(stripslashes($cats[cat_title]));
       $cat_desc = stripslashes($cats[cat_desc]);

        $display_block .= "<p><strong><a
        href=\"$_SERVER[PHP_SELF]?cat_id=$cat_id\">$cat_title</a></strong>
        <br>$cat_desc</p>";

       if ($_GET[cat_id] == $cat_id) {
           //preia produsele
           $get_items = "select id, item_title, item_price
           from store_items where cat_id = $cat_id
            order by item_title";
           $get_items_res = mysql_query($get_items) or die(mysql_error());

           if (mysql_num_rows($get_items_res) < 1) {
               $display_block = "<P><em>Sorry, no items in
                this category.</em></p>";
           } else {

               $display_block .= "<ul>";

               while ($items = mysql_fetch_array($get_items_res)) {
                   $item_id  = $items[id];
                   $item_title = stripslashes($items[item_title]);
                   $item_price = $items[item_price];

                   $display_block .= "<li><a
```

LISTINGUL 20.1. (*continuare*)

```
48:                     href=\"showitem.php?item_id=$item_id\">$item_title</a>
49:                     </strong> (\$$item_price)";
50:                 }
51:
52:                 $display_block .= "</ul>";
53:             }
54:         }
55:     }
56: }
57: ?>
58: <HTML>
59: <HEAD>
60: <TITLE>My Categories</TITLE>
61: </HEAD>
62: <BODY>
63 :<?php echo $display_block; ?>
64: </BODY>
65: </HTML>
```

Dacă ne gândim la dimensiunile scripturilor pe care le-ați văzut în capitolul 18, aceste 65 de linii perfect funcționale reprezintă o schimbare binevenită. Pe liniile 3–5, este deschisă conexiunea la baza de date deoarece, indiferent de acțiunea pe care scriptul va trebui să o execute — afișarea categoriilor sau afișarea produselor dintr-o anumită categorie — folosirea bazei de date este necesară.

Pe liniile 7–8, este începută construirea șirului de caractere `$display_block`, adăugându-i-se câteva informații de bază, respectiv titlul paginii. Pe linia 15 sunt verificate categoriile; dacă nu există nici o categorie în tabelă, este afișat un mesaj pentru utilizator și scriptul se oprește aici. Pe de altă parte, dacă sunt găsite categorii, scriptul trece mai departe la linia 19, pe care este începută o buclă `while` pentru extragerea informațiilor.

În cadrul blocului de cod al buclei `while`, pe liniile 20–22 este preluat ID-ul, titlul și descrierea fiecărei categorii. Sunt efectuate operațiuni cu șiruri de caractere pentru a ne asigura că în text nu există caractere de eludare și că numele categoriei este scris cu litere mari, din motive estetice. Pe liniile 24–26 informația despre categorii, inclusiv un link care face referire la aceeași pagină, este introdusă în șirul de caractere `$display_block`. Dacă utilizatorul dă clic pe linkul respectiv, el va fi direcționat către același script, însă, în șirul de interogare HTTP se va găsi ID-ul categoriei corespunzătoare linkului. Scriptul verifică această valoarea pe linia 28.

Dacă scriptului i-a fost pasată o valoare pentru `$cat_id` în urma faptului că utilizatorul dăduse clic pe un link corespunzător unei categorii în speranța de a vedea produsele din acea categorie, scriptul construiește și transmite o nouă interogare SQL (pe liniile 30–33) pentru a prelua produsele corespunzătoare categoriei din baza de date. Pe liniile 42–53 este verificată existența produselor și, apoi, este construit un șir de caractere conținând aceste produse care este adăugat șirului de caractere `$display_block`. O parte a acestui șir de caractere este reprezentată de linkuri către un script denumit `showitem.php`, pe care îl veți crea în secțiunea următoare.

După ce ajunge în acest punct, scriptul nu mai are nimic de făcut, aşa că afişează codul HTML şi valoarea şirului de caractere `$display_block`. Figura 20.1 arată rezultatul scriptului atunci când este accesat direct; sunt afişate doar informaţiile referitoare la categorii.

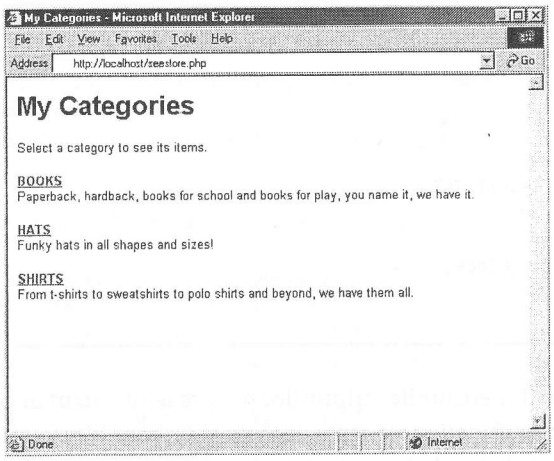


FIGURA 20.1. Categoriile disponibile în magazinul virtual.

În figura 20.2, puteţi vedea ce se întâmplă atunci când un utilizator dă clic pe linkul HATS: scriptul strânge toate produsele asociate cu această categorie şi le afişează pe ecran. Utilizatorul mai poate, totuşi, să sară la o altă categorie din aceeaşi pagină şi, dacă face acest lucru, îi vor fi afişate toate produsele din această nouă categorie.

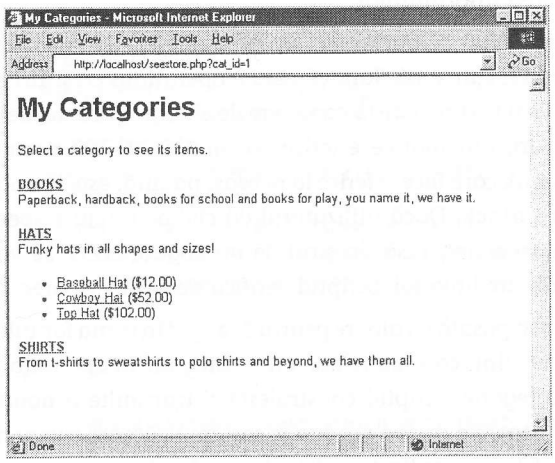


FIGURA 20.2. Produsele dintr-o anumită categorie.

Ultima piesă a acestui puzzle este crearea paginii în care să fie afişate informaţiile despre câte un produs.

Afişarea produselor

Pagina de afişare a produselor din acest capitol nu va face decât să afişeze toate informaţiile referitoare la câte un produs. În capitolul următor, veţi adăuga câteva linii care vor crea un buton cu ajutorul căruia un produs să poată fi „adăugat în coş". Aşadar, deocamdată, vom presupune că această pagină este similară unui catalog pe suport de hârtie.

LISTINGUL 20.2. Scriptul care permite vizualizarea informaţiilor.

```
<?php
//conectare la baza de date
$conn = mysql_connect("localhost", "joeuser", "somepass")
    or die(mysql_error());
mysql_select_db("testDB",$conn)  or die(mysql_error());

$display_block = "<h1>My Store - Item Detail</h1>";

//validarea produsului
$get_item = "select c.cat_title, si.item_title, si.item_price,
si.item_desc, si.item_image from store_items as si left join
store_categories as c on c.id = si.cat_id where si.id = $_GET[item_id]";

$get_item_res = mysql_query($get_item) or die (mysql_error());

if (mysql_num_rows($get_item_res) < 1) {
   //produsul nu este valid
   $display_block .= "<P><em>Invalid item selection.</em></p>";
} else {
   //produs valid, preia informatiile despre acesta
   $cat_title = strtoupper(stripslashes(
        mysql_result($get_item_res,0,'cat_title')));
   $item_title = stripslashes(mysql_result($get_item_res,0,'item_title'));
   $item_price = mysql_result($get_item_res,0,'item_price');
   $item_desc = stripslashes(mysql_result($get_item_res,0,'item_desc'));
   $item_image = mysql_result($get_item_res,0,'item_image');

   //creaza "dara de firmituri" - breadcrumb trail
   $display_block .= "<P><strong><em>You are viewing:</em><br>
   <a href=\"seestore.php?cat_id=$cat_id\">$cat_title</a>
    &gt; $item_title</strong></p>

   <table cellpadding=3 cellspacing=3>
   <tr>
   <td valign=middle align=center><img src=\"$item_image\"></td>
   <td valign=middle><P><strong>Description:</strong><br>$item_desc</p>
   <P><strong>Price:</strong> \$$item_price</p>";

   //preia culorile disponibile
   $get_colors = "select item_color from store_item_color where
    item_id = $item_id order by item_color";
   $get_colors_res = mysql_query($get_colors) or die(mysql_error());

   if (mysql_num_rows($get_colors_res) > 0) {

       $display_block .= "<P><strong>Available Colors:</strong><br>";

```

LISTINGUL 20.2. *(continuare)*

```
48:         while ($colors = mysql_fetch_array($get_colors_res)) {
49:             $item_color = $colors['item_color'];
50:
51:             $display_block .= "$item_color<br>";
52:         }
53:     }
54:
55:     //preia masurile disponibile
56:     $get_sizes = "select item_size from store_item_size where
57:         item_id = $item_id order by item_size";
58:     $get_sizes_res = mysql_query($get_sizes) or die(mysql_error());
59:
60:     if (mysql_num_rows($get_sizes_res) > 0) {
61:
62:         $display_block .= "<P><strong>Available Sizes:</strong><br>";
63:
64:         while ($sizes = mysql_fetch_array($get_sizes_res)) {
65:             $item_size = $sizes['item_size'];
66:
67:             $display_block .= "$item_size<br>";
68:         }
69:     }
70:
71:     $display_block .= "
72:     </td>
73:     </tr>
74:     </table>";
75:
76: }
77: ?>
78: <HTML>
79: <HEAD>
80: <TITLE>My Store</TITLE>
81: </HEAD>
82: <BODY>
83: <?php echo $display_block; ?>
84: </BODY>
85: </HTML>
```

Pe liniile 3–5 este deschisă conexiunea la baza de date deoarece întreg conţinutul acestei pagini este bazat pe informaţiile din baza de date. Pe linia 7, începem construirea şirului de caractere `$display_block`, în care adăugăm câteva informaţii de bază — titlul paginii.

Pe liniile 10–14 creăm şi trimitem interogarea pentru extragerea categoriei şi a informaţiilor despre produs. Această interogare reprezintă o joncţiune între tabele. În loc să selectăm informaţiile despre produs dintr-o tabelă şi apoi să emitem o nouă interogare pentru a găsi numele categoriei din care face parte, am folosit o interogare care uneşte tabelele pe baza câmpului ce conţine ID-ul categoriei pentru a extrage, în acelaşi timp, datele despre produs şi numele categoriei.

Pe linia 16 este verificat rezultatul; dacă nu există nici un produs în baza de date care să corespundă interogării, este afişat un mesaj pentru utilizator şi scriptul se încheie aici. Pe de altă parte, dacă informaţiile despre produs sunt găsite, scriptul trece mai departe şi culege informaţiile pe liniile 21–26.

Pe liniile 29–31, creaţi ceea ce se numeşte o „dâră de firimituri" (breadcrumb trail). Acesta este un element de navigare prin site pe care utilizatorul îl poate folosi ca să se întoarcă la obiectele de nivel mai înalt din cadrul arhitecturii ierarhice a site-ului. Aceste cuvinte pompoase înseamnă, de fapt „afişează un link pe care utilizatorul să dea clic ca să revină la lista categoriilor".

Pe liniile 33–37, continuăm să adăugăm informaţii despre produs în şirul de caractere `$display_block`, creând un tabel în care să fie afişate aceste informaţii. Folosim valorile culese pe liniile 21–26 pentru a crea un link sub forma unei imagini, să afişăm descrierea produsului şi să afişăm preţul acestuia. În acest moment, nu mai lipsesc decât culoarea şi dimensiunile, aşa că liniile 39–53 selectează şi afişează toate culorile asociate cu acest produs, iar liniile 55–69 extrag dimensiunile asociate cu acest produs.

Liniile 71–74 încheie şirul `$display_block` şi, din moment ce scriptul nu mai are nimic de făcut, codul HTML este afişat împreună cu valoarea şirului de caractere `$display_block`. Figura 20.3 arată rezultatul scriptului atunci când e selectată pălăria de cowboy din categoria pălăriilor. Bineînţeles, ceea ce va fi afişat pe calculatorul dumneavoastră o să arate un pic diferit, dar înţelegeţi ideea.

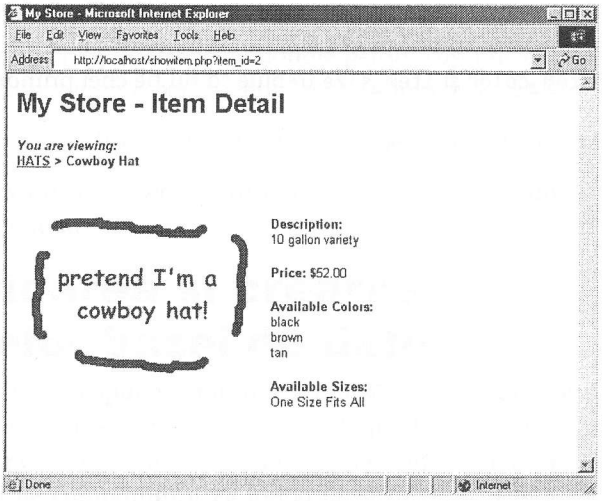


FIGURA 20.3. Pagina de produs corespunzătoare pălăriei de cowboy.

Asta este tot este ne trebuie pentru a crea o pagină simplă de afişare a unui produs. În capitolul următor, veţi modifica acest script astfel încât produsele să poată fi adăugate în coşul virtual de cumpărături.

Rezumat

În acest capitol practic, aţi aplicat cunoştinţele dumneavoastră de PHP şi MySQL pentru a crea interfaţa unui magazin virtual. Aţi învăţat cum să creaţi tabelele bazei de date şi scripturile pentru vizualizarea categoriilor, a listelor de produse şi a fiecărui produs în parte.

Atelier

Atelierul este conceput să anticipeze posibile întrebări, să vă ajute să recapitulaţi cele învăţate şi să începeţi să vă puneţi cunoştinţele în practică.

Test

1. Ce funcţie PHP a fost folosită pentru a converti în majuscule şirurile corespunzătoare titlurilor categoriilor?
2. De ce tabelele `store_item_size` şi `store_item_color` nu conţin nici o cheie primară sau unică?

Soluţii

1. `strtoupper()`
2. Se poate presupune că veţi avea produse cu mai multe culori şi cu mai multe dimensiuni. De aceea, `item_id` nu este o cheie primară sau unică. De asemenea, mai multe produse pot avea aceleaşi culori sau dimensiuni, aşa că `item_color` şi `item_size` trebuie să nu fie chei primare sau unice.

CAPITOLUL 21

Crearea unui mecanism pentru un coş de cumpărături virtual

În acest capitol practic, veţi integra un mecanism pentru un „coş de cumpărături" (shopping cart) şi o procedură de „plată la casă" (checkout procedure) pentru interfaţa magazinului virtual pe care aţi creat-o în capitolul precedent. Vi se vor arăta metodele de creare a tabelelor necesare din baza de date, precum şi scripturile pentru adăugarea şi ştergerea produselor din coş. Şi de această dată, exemplele folosite în acest capitol reprezintă o cale de îndeplinire a acestei sarcini, dar numărul alternativelor este infinit. Scopul acestor exemple este să vă ofere un punct de pornire şi nu o metodă finală pentru construirea unui magazin virtual.

În acest capitol, veţi învăţa:

- cum să creaţi tabelele relaţionale pentru coşul de cumpărături şi pentru plata produselor pentru magazinul dumneavoastră virtual;
- cum să creaţi scripturile care să adauge şi să elimine produsele din coş;
- câteva metode pentru procesarea tranzacţiilor, şi cum să creaţi secvenţa de plată la casă.

Planificarea şi crearea tabelelor bazei de date

Din moment ce scopul acestui capitol este să îi ofere utilizatorului o modalitate prin care acesta să selecteze şi să comande produse, vă puteţi imagina cam ce tabele vor fi necesare — în primul şi în primul rând, veţi avea nevoie de o tabelă pentru coşul de cumpărături! Pe lângă acest tabel, veţi mai avea nevoie şi de o tabelă în care să stocaţi comenzile, precum şi o tabelă în care să fie stocate produsele cumpărate în cadrul fiecărei comenzi.

Următoarele instrucţiuni SQL au fost folosite pentru crearea a trei noi tabele, începând cu tabelul `store_shoppertrack`. Acesta va fi tabelul în care vom stoca produsele pe măsură ce ele sunt adăugate de către utilizatori în coşurile lor de cumpărături.

Apropo

Lungimile câmpurilor care au fost folosite la definirea acestor tabele au fost alese arbitrar pentru a permite o anumită gamă de date introduse de utilizator. Dumneavoastră puteți să modificați aceste lungimi ale câmpurilor astfel încât ele să se potrivească nevoilor dumneavoastră specifice.

```
mysql> create table store_shoppertrack (
    -> id int not null primary key auto_increment,
    -> session_id varchar (32),
    -> sel_item_id int,
    -> sel_item_qty smallint,
    -> sel_item_size varchar(25),
    -> sel_item_color varchar(25),
    -> date_added datetime
    -> );
Query OK, 0 rows affected (0.01 sec)
```

În acest tabel, singura cheie este câmpul `id` al înregistrării. Câmpul `session_id` nu poate fi unic; altminteri, utilizatorii nu ar putea comanda decât un singur produs din magazinul dumneavoastră, iar asta nu ar fi o practică benefică pentru afacere. Câmpul `session_id` identifică utilizatorul. Câmpurile `sel_*` reprezintă selecţiile efectuate de către utilizator: produsul selectat, cantitatea din respectivul produs care a fost selectată şi culoarea şi dimensiunea produsului selectat. În fine, mai există şi un câmp intitulat `date_added`. De multe ori, utilizatorii adaugă produse în coşul de cumpărături, dar nu finalizează procesul de cumpărare niciodată. Din acest motiv, în tabela dumneavoastră de urmărire vor apărea o mulţime de produse rătăcite, pe care aţi putea dori să le eliminaţi periodic. De exemplu, aţi putea şterge toate produsele care se găsesc în coşuri de cumpărături de mai mult de o săptămână — şi, iată, câmpul `date_added` se dovedeşte, în acest caz, foarte util.

Următorul tabel conţine informaţiile despre comenzi:

```
mysql> create table store_orders (
    -> id int not null primary key auto_increment,
    -> order_date datetime,
    -> order_name varchar (100),
    -> order_address varchar (255),
    -> order_city varchar (50),
    -> order_state char(2),
    -> order_zip varchar(10),
    -> order_tel varchar(25),
    -> order_email varchar(100),
    -> item_total float(6,2),
    -> shipping_total float(6,2),
    -> authorization varchar (50),
    -> status enum('processed', 'pending')
    -> );
Query OK, 0 rows affected (0.00 sec)
```

Singura cheie din tabela `store_orders` este câmpul `id`. Pentru ca lecţia aceast să nu devină excesiv de lungă, vom considera că adresa de facturare şi adres de livrare pentru un utilizator sunt una şi aceeaşi de fiecare dată şi că ace magazin vinde produse numai către adrese din Statele Unite. Este relativ simpl să adăugaţi încă un bloc de câmpuri pentru informaţiile referitoare la adresa d livrare, dacă veţi dori să o faceţi. De asemenea, acest tabel pleacă de la premis că dumneavoastră nu veţi stoca informaţii despre cardurile de credit — şi nu trebui să faceţi niciodată acest lucru, în afara cazului în care aveţi un sistem d

super-criptare a informaţiei şi un firewall de mare performanţă. Această tabelă este bazată pe ideea procesării în timp real a plăţilor cu cardurile de credit. Veţi învăţa câteva opţiuni pentru realizarea tranzacţiilor la sfârşitul acestei lecţii. Ultima tabelă este aceea în care vor fi stocate informaţiile despre produsele din fiecare comandă şi se numeşte `store_orders_items`:

```
mysql> create table store_orders_items (
    -> id int not null primary key auto_increment,
    -> order_id int,
    -> sel_item_id int,
    -> sel_item_qty smallint,
    -> sel_item_size varchar(25),
    -> sel_item_color varchar(25),
    -> sel_item_price float(6,2)
    -> );
Query OK, 0 rows affected (0.00 sec)
```

Câmpurile `sel_*` ar trebui să vă fie familiare — cu excepţia câmpului `sel_item_price`, ele sunt aceleaşi câmpuri care apar şi în tabela `store_shoppertrack`. Cheia primară a acestei tabele este câmpul `id`, în timp ce câmpul `order_id` este folosit pentru legarea fiecărei linii corespunzătoare unui produs cu înregistrarea respectivă din tabela store_orders. Câmpul `sel_item_price` este inclus aici şi nu preluat prin legătură din tabela `store_items`, pentru că preţul din înregistrarea corespunzătoare produsului se poate schimba. Dacă schimbaţi preţul unui produs, atunci, în cazul în care aţi lega înregistrările produselor vândute cu preţul din tabela produselor, produsul vândut va apărea cu preţul nou, adică nu cu suma efectivă pe care a plătit-o utilizatorul.

Având toate tabelele puse la punct, putem să mergem mai departe şi să începem să adăugăm un produs în coşul de cumpărături al utilizatorului.

Integrarea coşului de cumpărături în interfaţa magazinului virtual

În această secţiune, veţi aduce modificări scriptului `showitem.php` din capitolul 20, intitulat *Crearea unei interfeţe pentru un magazin virtual.* Scopul acestor modificări este acela de a transforma pagina cu informaţii despre produs într-o pagină care să conţină un formular pentru selectarea culorilor, dimensiunilor şi cantităţilor.

În scriptul original, inseraţi următoarea linie înainte de linia 2:

```
session_start();
```

Din moment ce elementele coşului de cumpărături vor fi accesate de către utilizator prin intermediul ID-ului de sesiune, trebuie iniţiată o sesiune. Următoarele schimbări nu au loc decât acolo unde era linia 37 din scriptul original, aşa că listingul 21.1 începe din acel loc.

LISTINGUL 21.1. Liniile noi din scriptul showitem.php.

```
37:        <P><strong>Price:</strong> \$$item_price</p>
38:        <form method=post action=\"addtocart.php\">";
39:    //preia culorile disponibile
40:    $get_colors = "select item_color from store_item_color where
41:         item_id = $item_id order by item_color";
42:    $get_colors_res = mysql_query($get_colors) or die(mysql_error());
43:
44:    if (mysql_num_rows($get_colors_res) > 0) {
45:
46:        $display_block .= "<P><strong>Available Colors:</strong>
47:        <select name=\"sel_item_color\">";
48:
49:        while ($colors = mysql_fetch_array($get_colors_res)) {
50:            $item_color = $colors['item_color'];
51:            $display_block .=
52:             "<option value=\"$item_color\">$item_color</option>";
53:        }
54:
55:        $display_block .= "</select>";
56:    }
57:
58:    //preia masurile disponibile
59:    $get_sizes = "select item_size from store_item_size where
60:         item_id = $item_id order by item_size";
61:    $get_sizes_res = mysql_query($get_sizes) or die(mysql_error());
62:
63:    if (mysql_num_rows($get_sizes_res) > 0) {
64:
65:        $display_block .= "<P><strong>Available Sizes:</strong>
66:        <select name=\"sel_item_size\">";
67:
68:        while ($sizes = mysql_fetch_array($get_sizes_res)) {
69:            $item_size = $sizes['item_size'];
70:            $display_block .= "
71:             <option value=\"$item_size\">$item_size</option>";
72:        }
73:
74:        $display_block .= "</select>";
75:    }
76:
77:    $display_block .= "
78:    <P><strong>Select Quantity:</strong>
79:    <select name=\"sel_item_qty\">";
80:
81:    for($i=1; $i<11; $i++) {
82:        $display_block .= "<option value=\"$i\">$i</option>";
83:    }
84:
```

LISTINGUL 21.1. (*continuare*)

```
 85:    $display_block .= "
 86:    </select>
 87:    <input type=\"hidden\" name=\"sel_item_id\" value=\"$_GET[item_id]\">
 88:    <P><input type=\"submit\" name=\"submit\" value=\"Add to Cart\"></p>
 89:    </form>
 90:     </td>
 91:    </tr>
 92:    </table>";
 93: }
 94: ?>
 95: <HTML>
 96: <HEAD>
 97: <TITLE>My Store</TITLE>
 98: </HEAD>
 99: <BODY>
100: <?php echo $display_block; ?>
101: </BODY>
102: </HTML>
```

Prima schimbare este în locul în care, în scriptul original, se găsea linia 37, acolo unde şirul de caractere `$display_block` este continuat pentru a include un element `<form>`. Acţiunea acestui formular este un script denumit `addtocart.php`, pe care îl veţi crea în secţiunea următoare.

Următoarea schimbare are loc pe linia 47, acolo unde şirul `$display_block` este continuat pentru a include marcajul de deschidere al unui element `<select>`, denumit `sel_item_color`. Pe liniile 51–52, culorile, în loc să fie pur şi simplu afişate pe ecran, sunt introduse în elementele `<option>` pentru ca utilizatorul să aleagă dintre ele. Linia 55 încheie elementul `<select>`.

Aceleaşi modificări sunt efectuate şi pentru dimensiunile produselor. Liniile 66–67 reflectă continuarea şirului `$display_block` pentru a include elementul `<select>`, denumit `sel_item_size`. Liniile 70–71 scriu dimensiunile în cadrul elementelor `<option>`, iar linia 74 închide elementul `<select>`.

Liniile 77–83 reprezintă adăugiri aduse scriptului. Aceste linii creează un element `<select>`, denumit `sel_item_qty`, cu ajutorul căruia utilizatorul va alege cantitatea pe care doreşte să o cumpere din respectivul produs. Linia 86 închide acest element `<select>`, iar linia 87 adaugă un câmp ascuns pentru `item_id`. Linia 88 adaugă butonul de transmitere a formularului, iar linia 89 încheie elementul `<form>`. Liniile ce urmează sunt neschimbate.

Atunci când vizualizaţi pălăria de cowboy cu ajutorul noii versiuni a scriptului `showitem.php`, veţi vedea imaginea din figura 21.1, reflectând adăugarea elementelor de tip formular.

Următorul pas este crearea scriptului `addtocart.php`.

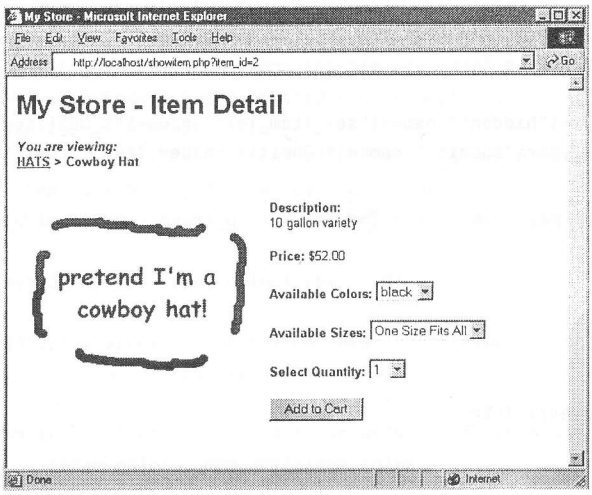


FIGURA 21.1. Noua pagină a produsului pălărie de cowboy.

Adăugarea de elemente în coşul dumneavoastră de cumpărături

Scriptul `addtocart.php` nu va face altceva decât să scrie în tabela `store_shppertrack` şi să redirecţioneze utilizatorul către pagina de vizualizare a coşului de cumpărături. Vom crea scriptul `addtocart.php` în listingul 21.2 şi, apoi, vom aborda scriptul `showcart.php` — care va afişa coşul de cumpărături — în secţiunea următoare.

LISTINGUL 21.2. Scriptul `addtocart.php`.

```
<?php
session_start();

//conectare la baza de date
$conn = mysql_connect("localhost", "joeuser", "somepass")
or die(mysql_error());
mysql_select_db("testDB",$conn)  or die(mysql_error());

if ($_POST[sel_item_id] != "") {
   //validarea produsului si preluarea numelui si a pretului acestuia
   $get_iteminfo = "select item_title from store_items
        where id = $_POST[sel_item_id]";
   $get_iteminfo_res = mysql_query($get_iteminfo)
        or die(mysql_error());

   if (mysql_num_rows($get_iteminfo_res) < 1) {
       //id invalid, se revine pe pagina
       header("Location: seestore.php");
       exit;
   } else {
       //preluare informatii
       $item_title =  mysql_result($get_iteminfo_res,0,'item_title');

```

LISTINGUL 21.2. *(continuare)*

```
25:         //adaugare informatii la tabela cart
26:         $addtocart = "insert into store_shoppertrack values
27:          ('', '$PHPSESSID', '$_POST[sel_item_id]', '$_POST[sel_item_qty]',
28:          '$_POST[sel_item_size]', '$_POST[sel_item_color]', now())";
29:
30:         mysql_query($addtocart);
31:
32:         //redirectionare la pagina showcart
33:         header("Location: showcart.php");
34:         exit;
35:
36:     }
37: } else {
38:     //trimite utilizatorul pe alta pagina din site
39:     header("Location: seestore.php");
40:     exit;
41: }
42: ?>
```

Linia 2 continuă sesiunea de utilizator, lucru important având în vedere că va trebui să preluaţi ID-ul sesiunii utilizatorului pentru a putea scrie în tabela `store_shoppertrack`. Liniile 5–7 iniţiază conexiunea la baza de date, iar linia 9 începe validarea acţiunilor.

Pe linia 9, scriptul verifică dacă este prezentă o valoare în `$_POST[sel_item_id]`, ceea ce ar însemna că utilizatorul a ajuns la acest script dinspre un formular. Dacă nu există o astfel de valoare, scriptul sare direct la linia 37, trimite utilizatorul către pagina principală pe linia 39 şi cu asta îşi încheie execuţia.

Pe de altă parte, în cazul în care `$_POST[sel_item_id]` conţine o valoare, următoarea acţiune este verificarea faptului că acea valoare este validă. Liniile 11–14 creează şi transmit o interogare SQL care culege titlul produsului selectat. Linia 16 verifică existenţa unui rezultat; dacă nu există nici un rezultat, utilizatorul este redirecţionat din nou, pe linia 18.

Dacă produsul selectat este valid, scriptul continuă cu linia 22 şi extrage valorile din setul de rezultate. Scriptul are acum suficiente informaţii pentru a adăuga produsul respectiv în tabela `store_shoppertrack`, lucru pe care îl şi face pe liniile 26–30.

După ce interogarea este trimisă, utilizatorul este direcţionat către scriptul `showcart.php`, care conţine toate elementele din coşul de cumpărături. Veţi crea acest script în secţiunea care urmează.

Vizualizarea coşului de cumpărături

Acum că puteţi adăuga produse în coşul de cumpărături, veţi dori să le şi vedeţi! Listingul 21.3 vă arată codul scriptului `showcart.php`.

LISTINGUL 21.3. Scriptul showcart.php.

```
<?php
session_start();
//conectare la baza de date
$conn = mysql_connect("localhost", "joeuser", "somepass")
    or die(mysql_error());
mysql_select_db("testDB",$conn)  or die(mysql_error());

$display_block = "<h1>Your Shopping Cart</h1>";

//se verifica prezena de produse in cod pe baza id-ului de sesiune
$get_cart = "select st.id, si.item_title, si.item_price, st.sel_item_qty,
st.sel_item_size, st.sel_item_color from store_shoppertrack as st
left join store_items as si on si.id = st.sel_item_id where
session_id = '$PHPSESSID'";

$get_cart_res = mysql_query($get_cart) or die(mysql_error());

if (mysql_num_rows($get_cart_res) < 1) {
   //se afiseaza un mesaj
   $display_block .= "<P>You have no items in your cart.
    Please <a href=\"seestore.php\">continue to shop</a>!</p>";

} else {
    //preia informatii si afiseaza continutul cosului de cumparaturi
    $display_block .= "
    <table celpadding=3 cellspacing=2 border=1 width=98%>
   <tr>
   <th>Title</th>
   <th>Size</th>
   <th>Color</th>
   <th>Price</th>
   <th>Qty</th>
   <th>Total Price</th>
   <th>Action</th>
   </tr>";

   while ($cart = mysql_fetch_array($get_cart_res)) {
       $id = $cart['id'];
       $item_title = stripslashes($cart['item_title']);
       $item_price = $cart['item_price'];
       $item_qty = $cart['sel_item_qty'];
       $item_color = $cart['sel_item_color'];
       $item_size = $cart['sel_item_size'];

       $total_price = sprintf("%.02f", $item_price * $item_qty);

       $display_block .= "<tr>
       <td align=center>$item_title <br></td>
       <td align=center>$item_size <br></td>
       <td align=center>$item_color <br></td>
       <td align=center>\$ $item_price <br></td>
       <td align=center>$item_qty <br></td>
       <td align=center>\$ $total_price</td>
       <td align=center><a href=\"removefromcart.php?id=$id\">remove</a></td>
       </tr>";
```

LISTINGUL 21.3. (*continuare*)

```
56:     }
57:
58:     $display_block .= "</table>";
59: }
60: ?>
61: <HTML>
62: <HEAD>
63: <TITLE>My Store</TITLE>
64: </HEAD>
65: <BODY>
66: <?php echo $display_block; ?>
67: </BODY>
68: </HTML>
```

Linia 2 continuă sesiunea de utilizator, lucru foarte important deoarece veţi avea nevoie să coroboraţi ID-ul de sesiune al utilizatorului cu înregistrarea din tabela `store_shoppertrack`. Liniile 4–6 realizează conexiunea la baza de date, iar linia 8 începe şirul de caractere `$display_block`, adăugându-i un titlu pentru pagină.

Liniile 10–14 reprezintă o interogare cu joncţiune între tabele, cu ajutorul căreia sunt extrase produsele salvate de utilizator. Câmpurile `id`, `sel_item_qty`, `sel_item_size` şi `sel_item_color` sunt extrase din tabela `store_shoppertrack`, iar câmpurile `item_title` şi `item_price` sunt extrase din tabela `store_items`, pe baza informaţiei comune din câmpul `id` al tabelei `store_items` şi sel_item_id din tabela `store_shoppertrack`. Cu alte cuvinte, în loc să afişăm valoarea 2 pentru produsul selectat, afişăm titlul respectivului produs, adică Cowboy Hat. Linia 16 transmite interogarea, iar linia 18 verifică existenţa rezultatelor.

Dacă nu există rezultate, utilizatorul nu are nici un produs în tabela `store_shoppertrack` aşa ca, în şirul de caractere `$display_block`, este scris un mesaj, scriptul afişează acest mesaj şi îşi încheie execuţia.

Dacă există într-adevăr rezultate, pe liniile 25–25 este creat începutul unui tabel HTML, cu coloane definite pentru fiecare dintre informaţiile conţinute în coş (şi, apoi, încă vreo câteva). Linia 37 începe bucla `while` care extrage fiecare produs din tabela `store_shoppertrack`, iar această buclă continuă până pe linia 56, afişând informaţia în celula corespunzătoare din tabelul HTML.

Pe linia 54, vedeţi că este creat un link spre un script de eliminare a produsului, pe care îl veţi crea în secţiunea următoare. Linia 58 închide tabela, iar scriptul îşi termină execuţia afişând codul HTML în browser pe liniile 59–68.

Acum, haideţi să ne întoarcem la pagina unui produs şi să îl adăugăm în coşul de cumpărături. După ce elementele sunt adăugate în tabela `store_shoppertrack`, ar trebui să fiţi redirecţionat către pagina `showcart.php`, iar noul produs selectat ar trebui să fie afişat acolo, împreună cu celelalte produse selectate anterior. Figura 21.1 arată coşul meu de cumpărături după adăugarea câtorva produse.

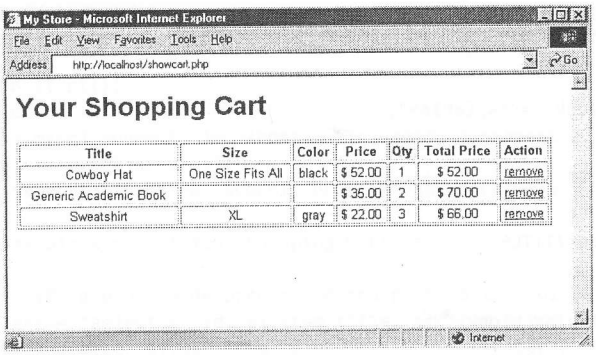


FIGURA 21.2. Produsele adăugate în coşul de cumpărături.

Următoarea etapă este reprezentată de crearea scriptului `removefromthecart.php` care va elimina produse din coşul de cumpărături.

Eliminarea produselor din coşul de cumpărături

Scriptul `removefromthecart.php` este relativ scurt, deoarece nu face altceva decât să emită o interogare şi să redirecţioneze utilizatorul către un alt script. Inevitabil, utilizatorii vor dori să şteargă unele produse din coşurile lor de cumpărături, iar scriptul `removefromthecart.php` le permite să facă exact acest lucru. Listingul 21.4 arată întregul script.

LISTINGUL 21.4. Scriptul `removefromthecart.php`.

```
<?php
session_start();
//conectare la baza de date
$conn = mysql_connect("localhost", "joeuser", "somepass")
    or die(mysql_error());
mysql_select_db("testDB",$conn)  or die(mysql_error());

if ($_GET[id] != "") {
   $delete_item = "delete from store_shoppertrack where
    id = $_GET[id] and session_id = '$PHPSESSID'";
   mysql_query($delete_item) or die(mysql_error());

    //redirectionare la pagina showcart
   header("Location: showcart.php");
   exit;

} else {
   //trimite utilizatorul pe o alta pagina
   header("Location: seestore.php");
   exit;
}
?>
```

Linia 2 continuă sesiunea utilizatorului deoarece trebuie să coroboraţi ID-ul acestei sesiuni cu înregistrările din tabela `store_shoppertrack`. Liniile 4–6 iniţiază conexiunea la baza de date, iar linia 8 verifică existenţa unei valori în `$_GET[id]`. Dacă nu există o valoare în `$_GET[id]`, înseamnă utilizatorul nu a dat clic pe un link din pagina de vizualizare a coşului de cumpărături şi, de aceea, este trimis către pagina principală, pe linia 19.

Dacă există o valoare în `$_GET[id]`, atunci este emisă o interogare SQL (liniile 9–11), iar utilizatorul este redirecţionat către scriptul `showcart.php` (pe linia 14), unde produsul şters nu ar trebui să mai apară. Încercaţi şi vedeţi dacă se întâmplă aşa!

Metode de plată şi secvenţa de plată la casa virtuală

Există câteva metode atunci când vine vremea pentru plata cumpărăturilor din coşul de cumpărături. Metoda „corectă" depinde de dumneavoastră şi de compania dumneavoastră — deschiderea unui cont de comerciant la o instituţie bancară necesită, de cele mai multe ori, să aveţi o autorizaţie de funcţionare, un permis de distribuţie de produse precum şi alte documente care să demonstreze că reprezentaţi o firmă legitimă. Dacă sunteţi pur şi simplu o persoană fizică şi aveţi de vânzare câteva produse, s-ar putea să nu vreţi să faceţi rost de atâta documentaţie. Oricare ar fi situaţia, aveţi la dispoziţie o serie de opţiuni.

Indiferent de modalitatea de plată, un lucru este sigur: dacă transmiteţi informaţii despre carduri de credit prin intermediul Web-ului, trebuie să faceţi acest lucru printr-o conexiune de tip SSL. Obţinerea unui certificat SSL şi instalarea acestuia pe sistemul dumneavoastră sunt discutate în capitolul 27, *Configurarea unui server Web securizat*. Nu este nevoie să folosiţi conexiunea securizată pe tot parcursul vizitei utilizatorului, ci doar în momentul în care sunt preluate de la acesta informaţii cu caracter sensibil, cum este şi cazul formularului de plată.

Crearea unui formular de plată

În acest moment, după ce aţi citit această carte până aici, ar trebui să fiţi versat în ceea ce priveşte crearea unui formular simplu. La începutul acestui capitol, am creat o tabelă denumită `store_orders` care conţine câmpurile ce vor sta şi la baza formularului dumneavoastră:

- `order_name`
- `order_address`
- `order_city`
- `order_state`
- `order_zip`

- `order_tel`
- `order_email`

În plus, formularul dumneavoastră va mai avea nevoie şi de câmpuri pentru numărul cadrului de credit, al datei expirării acestuia, precum şi pentru numele de pe cardul de credit. Un alt element care ar trebui adăugat este un tabel care să recapituleze conţinutul coşului de cumpărături împreună cu un subtotal pentru fiecare produs, astfel încât utilizatorul să îşi dea seama pentru ce anume plăteşte şi cât va costa, aproximativ, comanda. De asemenea, în acest moment al secvenţei de plată, ar trebui să furnizaţi detalii despre opţiunile de livrare pe care le puneţi la dispoziţie. Taxele de transport şi vânzare vor fi calculate în etapa următoare.

Din momentul în care utilizatorul dă clic pe butonul de transmitere a acestui formular, secvenţa de plată depinde de metoda de plată pe care o folosiţi. Secţiunea următoare parcurge etapele de bază şi oferă sugestii asupra diverselor metode de procesare a plăţilor.

Efectuarea acţiunilor privind plata produselor

Dacă aţi obţinut un cont de comerciant prin intermediul băncii dumneavoastră, puteţi folosi un serviciu de plată în timp real cum este PayFlo Pro de la VeriSign. PHP are un set de funcţii predefinite care, atunci când sunt folosite împreună cu bibliotecile PayFlo Pro de la VeriSign, vă permit să creaţi un script foarte simplu care să proceseze tranzacţia cu cardul de credit. Puteţi să aflaţi mai multe despre PayFlo Pro de pe site-ul Web al companiei VeriSign: `http://www.verisign.com/products/payflow/pro/index.html`. Secţiunea din Manualul de PHP care se referă la funcţiile de lucru cu PayFlo se găseşte la adresa: `http://www.php.net/manual/ro/ref.pfpro.php`.

Produsul companiei VeriSign este unul dintre portalurile de procesare a tranzacţiilor care există şi este folosit de comercianţi. Banca dumneavoastră vă va oferi, de cele mai multe ori, o listă a sistemelor pe care preferă să îi folosiţi. Dacă nu respectaţi lista furnizorilor agreaţi de bancă, asiguraţi-vă că aţi verificat cu mare grijă furnizorul ales pentru a evita orice întârzieri în debitarea conturilor dumneavoastră, şi pentru a vă asigura că aţi ales o varianta cea mai profitabilă.

După ce aţi selectat procesorul de tranzacţie, scriptul dumneavoastră de plată ar trebui să urmeze un curs bazat pe indicaţiile de mai jos.

1. Însumaţi subtotalurile corespunzătoare fiecărui produs, adăugaţi taxele şi adăugaţi costul transportului. Astfel, obţineţi suma totală a cărei plată de pe cardul de credit trebuie autorizată.
2. Efectuaţi procedura de autorizare a plăţii de pe cardul de credit, folosind suma totală.
3. Veţi primi un rezultat semnificând fie reuşita, fie eşecul rutinei de procesare a cardului de credit. Dacă răspunsul este acela de eşec, afişaţi utilizatorului

un mesaj şi încheiaţi tranzacţia. Dacă răspunsul este acela de succes, continuaţi cu pasul 4.

4. Scrieţi informaţiile de bază despre comandă într-o tabelă precum `store_orders`, incluzând codul de autorizare pe care l-aţi primit de la sistemul de procesare a tranzacţiilor în urma unei autorizări reuşite. Obţineţi valoarea `id` a acestei înregistrări, folosind funcţia `mysql_insert_id()`.

5. Pentru fiecare produs din coşul de cumpărături care este legat de utilizatorul curent, inseraţi o înregistrare în tabela `store_orders_itemmap`. Fiecare înregistrare va face referire, în cadrul câmpului `order_id`, la valoarea câmpului `id` din tabela `store_orders` pe care aţi extras-o în etapa precedentă.

6. Ştergeţi produsele din coşul de cumpărături al utilizatorului curent.

7. Afişaţi comanda împreună cu codul de autorizare în locul în care se aflase formularul de introducere a informaţiilor despre cardul de credit, astfel încât utilizatorul să poată imprima această pagină şi să o păstreze în loc de chitanţă. Puteţi, de asemenea, să trimiteţi aceste informaţii utilizatorului prin e-mail.

Fiecare dintre etapele de mai sus — cu excepţia obţinerii efective a codului de autorizare — sunt la fel de simple ca toate etapele pe care le-aţi implementat pe parcursul acestei cărţi şi nu există nici un motiv pentru care ar trebui să complicaţi lucrurile mai mult decât este necesar!

Rezumat

În acest capitol practic, aţi aplicat cunoştinţele dumneavoastră de PHP şi MySQL pentru a integra un coş de cumpărături în interfaţa magazinului virtual pe care aţi creat-o în capitolul precedent. Printre altele, aţi creat tabelele bazei de date, aţi modificat pagina cu detaliile despre produs şi aţi scris câteva scripturi noi pentru adăugarea şi eliminarea de produse din coşul de cumpărături virtual.

Atelier

Atelierul este conceput să anticipeze posibile întrebări, să vă ajute să recapitulaţi cele învăţate şi să începeţi să vă puneţi cunoştinţele în practică.

Test

1. Atunci când eliminaţi un produs din coşul de cumpărături virtual, de credeţi că interogarea validează ID-ul de sesiune al utilizatorului folosind înregistrarea din baza de date?

2. Din ce motiv nu stocăm preţul într-un câmp ascuns atunci când adăugăm produse în coşul de cumpărături?

Soluţii

1. Utilizatorii ar trebui să nu poată să şteargă decât produsele din coşul propriu.

2. Dacă am stoca preţul într-un câmp ascuns, un utilizator rău voitor ar putea să schimbe respectiva valoare înainte de a transmite formularul, astfel, scriind preţul dorit de el în tabela `store_shoppertrack` şi nu preţul real.

CAPITOLUL 22
Crearea unui calendar simplu

Acest capitol va combina lecţiile pe care le-aţi învăţat până în acest moment despre limbajul PHP şi despre construirea unor mici aplicaţii. În acest capitol, veţi continua să învăţaţi astfel de lucruri în contextul creării unui mic calendar.

În acest capitol, veţi învăţa cum să construiţi:

- un script care să creeze un mic calendar;
- o bibliotecă de clase care să genereze liste derulante în formularele dumneavoastră HTML.

Construirea unui calendar simplu

Veţi folosi funcţiile de operare cu date calendaristice şi ore pe care le-aţi învăţat în capitolul 8, intitulat *Lucrul cu tipurile şir de caractere, dată şi oră*, pentru a construi un calendar ce va afişa datele din orice lună cuprinsă între 1980 şi 2010. Utilizatorul va avea posibilitatea să selecteze atât luna cât şi anul folosind meniuri derulante, iar datele corespunzătoare lunii selectate vor fi organizate în ordinea zilelor săptămânii. Vom lucra cu două variabile — una pentru lună şi alta pentru an — pe care le va furniza utilizatorul. Aceste informaţii vor fi folosite pentru a construi un *timestamp* bazat pe prima zi a lunii definite. Dacă datele introduse de utilizator vor fi absente sau invalide, valorile implicite vor reprezenta prima zi a lunii curente.

Verificarea datelor introduse de utilizator

Atunci când un utilizator deschide aplicaţia Calendar pentru prima oară, el o va face fără a transmite vreo informaţie. De aceea, va trebui să ne asigurăm că scriptul este în stare să funcţioneze chiar şi atunci când variabilele pentru lună şi an nu sunt definite. Am putea folosi în acest scop funcţia `isset()`, care returnează valoarea false dacă variabila care îi este pasată nu a fost definită. Totuşi, haideţi să folosim funcţia `checkdate()`, care nu numai că va sesiza dacă variabila nu există, dar va şi efectua o operaţiune utilă asupra ei, şi anume va stabili validitatea ei ca dată calendaristică. Listingul 22.1 arată fragmentul de cod care verifică variabilele `month` (lună) şi `year` (an) care provin dintr-un formular HTML şi construieşte un *timestamp* pe baza lor.

LISTINGUL 22.1. Verificarea datelor introduse de utilizator pentru scriptul Calendar.

```
 1: <?php
 2: if (!checkdate($_POST['month'], 1, $_POST['year'])) {
 3:     $nowArray = getdate();
 4:     $month = $nowArray['mon'];
 5:     $year = $nowArray['year'];
 6: } else {
 7:     $month = $_POST['month'];
 8:     $year = $_POST['year'];
 9: }
10: $start = mktime (12, 0, 0, $month, 1, $year);
11: $firstDayArray = getdate($start);
12: ?>
```

Listingul 22.1 este un fragment dintr-un script mai mare, aşa că el, în sine, nu produce nici un rezultat afişat. În cadrul instrucţiunii `if` de pe linia 2, folosim funcţia `checkdate()` pentru a verifica dacă luna şi anul au fost introduse în formular. Dacă aceste valori nu au fost definite, funcţia `checkdate()` returnează valoarea `false` pentru că sistemul nu poate să genereze o dată calendaristică validă pe baza unor argumente `month` şi `year` nedefinite. Această abordare are un avantaj suplimentar datorită faptului că ne asigură că date transmise de utilizator reprezintă, cu adevărat, o dată validă.

Dacă data nu este validă, pe linia 3 folosim funcţie `getdate()` pentru a crea un vector asociativ bazat pe data şi ora curentă. Apoi, atribuim noi valori variabilelor `$month` şi `$year`, folosind elementele `mon` şi `year` ale vectorului (pe liniile 4 şi 5). Dacă utilizatorul a introdus, totuşi, valori valide în formular, atribuim aceste valori variabilelor `$month` şi `$year` astfel încât să nu ne atingem de valorile originale stocate în vectorul superglobal `$_POST`.

Acum că suntem siguri că avem date valide în variabilele `$month` şi `$year`, putem folosi funcţia `mktime()` pentru a crea un timestamp pentru prima zi a lunii (pe linia 10). Vom avea nevoie de informaţii despre acest timestamp mai încolo, aşa că pe linia 11, creăm o variabilă denumită `$firstDayArray` în care vom stoca vectorul asociativ returnat de funcţia `getdate()` pe baza timestampului tocmai obţinut.

Construirea formularului HTML

Trebuie să creăm o interfaţă prin intermediul căreia utilizatorii să poată solicita vizualizarea datelor despre o anumită lună dintr-un anumit an. În acest scop, vom folosi elemente de tip `SELECT`. Deşi le-am putea codifica pe acestea static în HTML trebuie să ne asigurăm că listele derulante vor afişa implicit luna selectată într-un orice moment, aşa că vom crea aceste liste derulante în mod dinamic, adăugând câte un atribut `SELECT` în elementul `OPTION` atunci când este nevoie. Formularul este generat în listingul 22.2.

LISTINGUL 22.2. Construirea formularului HTML pentru scriptul Calendar.

```
1: <?php
2: if (!checkdate($_POST['month'], 1, $_POST['year'])) {
3:     $nowArray = getdate();
4:     $month = $nowArray['mon'];
5:     $year = $nowArray['year'];
6: } else {
7:     $month = $_POST['month'];
8:     $year = $_POST['year'];
9: }
10: $start = mktime (12, 0, 0, $month, 1, $year);
11: $firstDayArray = getdate($start);
12: ?>
13: <html>
14: <head>
15: <title><?php echo "Calendar:".$firstDayArray['month']."
16:  ".$firstDayArray['year'] ?></title>
17: <head>
18: <body>
19: <form method="post" action="<?php echo "$_SERVER[PHP_SELF]"; ?>">
20: <select name="month">
21: <?php
22: $months = Array("January", "February", "March", "April", "May",
23: "June", "July", "August", "September", "October", "November", "December");
24: for ($x=1; $x <= count($months); $x++) {
25:     echo"<option value=\"$x\"";
26:     if ($x == $month) {
27:         echo " SELECTED";
28:     }
29:     echo ">".$months[$x-1]."";
30: }
31: ?>
32: </select>
33: <select name="year">
34: <?php
35: for ($x=1980; $x<=2010; $x++) {
36:     echo "<option";
37:     if ($x == $year) {
38:         echo " SELECTED";
39:     }
40:     echo ">$x";
41: }
42: ?>
43: </select>
44: <input type="submit" value="Go!">
45: </form>
46: </body>
47: </html>
```

O dată creat *timestamp*-ul `$start` şi vectorul `$firstDayArray` (liniile 2–11), haideţi să scriem codul HTML al paginii. Observaţi că folosim vectorul `$firstDayArray` pentru a adăuga luna şi anul în elementul `TITLE` al paginii HTML pe liniile 15 şi 16.

Linia 19 reprezintă începutul formularului nostru. Pentru a crea elementul `SELECT` pentru lista derulantă a lunilor, trecem înapoi în modul PHP pe linia 21 pentru a scrie, pe rând, fiecare marcaj `OPTION`. Mai întâi creăm un vector denumit `$months` pe liniile 22 şi 23, care conţine numele celor 12 luni. După aceasta, parcurgem iterativ acest vector creând câte un marcaj `OPTION` pentru fiecare nume (pe liniile 24–30). Această metodă ar fi, probabil, una multe prea complicată pentru scrierea unui simplu element `SELECT`, dar, astfel, putem să verificăm dacă `$x` (variabila contor pentru instrucţiunea `for`) corespunde valorii din variabila `$month` (luna selectată momentan), lucru pe care îl facem pe linia 26. Dacă `$x` şi `$month` sunt echivalente, adăugăm şirul de caractere `SELECTED` în interiorul marcajului `OPTION`, asigurându-ne că luna corectă va fi selectată automat atunci când se încarcă pagina. Folosim o tehnică similară pentru a scrie lista derulantă corespunzătoare anilor, pe liniile 35–41. La final, trecem din nou în modul HTML şi creăm butonul de transmitere a datelor pe linia 44.

Acum avem un formular care îşi poate trimite sieşi parametri `month` şi `year` şi care, automat, va selecta luna şi anul curent, sau luna şi anul care au fost selectate anterior. Dacă salvaţi acest listing sub numele `listing22.2.php`, îl amplasaţi în directorul rădăcină al serverului dumneavoastră Web şi îl accesaţi cu ajutorul browserului, ar trebui să vedeţi ceva asemănător cu figura 22.1 (luna şi anul ar putea fi diferite la dumneavoastră).

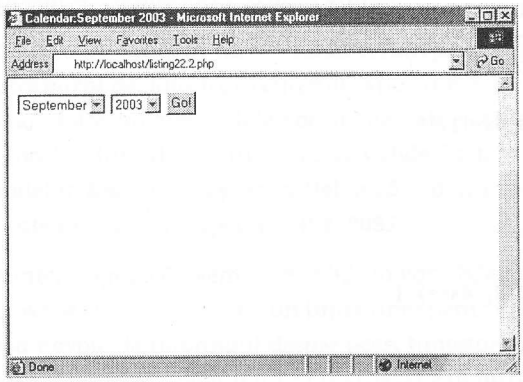


FIGURA 22.1. Formularul scriptului Calendar.

Crearea tabelului pentru calendar

Acum putem să adăugăm un tabel şi să îl populăm cu datele lunii alese. Facem acest lucru în listingul 22.3, care reprezintă scriptul complet al aplicaţiei.

LISTINGUL 22.3. Scriptul Calendar complet.

```
1: <?php
2: define("ADAY", (60*60*24));
3:  if (!checkdate($_POST['month'], 1, $_POST['year'])) {
4:     $nowArray = getdate();
5:     $month = $nowArray['mon'];
6:     $year = $nowArray['year'];
7:  } else {
```

LISTINGUL 22.3. (*continuare*)

```
8:     $month = $_POST['month'];
9:     $year = $_POST['year'];
10:  }
11: $start = mktime (12, 0, 0, $month, 1, $year);
12: $firstDayArray = getdate($start);
13: ?>
14: <html>
15: <head>
16: <title><?php echo "Calendar:".$firstDayArray['month']."
17:  ".$firstDayArray['year'] ?></title>
18: <head>
19: <body>
20: <form method="post" action="<?php echo "$_SERVER[PHP_SELF]"; ?>">
21: <select name="month">
22: <?php
23: $months = Array("January", "February", "March", "April", "May",
24: "June", "July", "August", "September", "October", "November", "December");
25: for ($x=1; $x <= count($months); $x++) {
26:     echo"<option value=\"$x\"";
27:     if ($x == $month) {
28:         echo " SELECTED";
29:     }
30:     echo ">".$months[$x-1]."";
31: }
32: ?>
33: </select>
34: <select name="year">
35: <?php
36: for ($x=1980; $x<=2010; $x++) {
37:     echo "<option";
38:     if ($x == $year) {
39:         echo " SELECTED";
40:     }
41:     echo ">$x";
42: }
43: ?>
44: </select>
45: <input type="submit" value="Go!">
46: </form>
47: <br>
48: <?php
49: $days = Array("Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat");
50: echo "<TABLE BORDER=1 CELLPADDING=5><tr>\n";
51: foreach ($days as $day) {
52:     echo "<TD BGCOLOR=\"#CCCCCC\" ALIGN=CENTER><strong>$day</strong></td>\
n";
53: }
54: for ($count=0; $count < (6*7); $count++) {
55:     $dayArray = getdate($start);
56:     if (($count % 7) == 0) {
57:         if ($dayArray['mon'] != $month) {
58:             break;
```

LISTINGUL 22.3. (*continuare*)

```
59:          } else {
60:            echo "</tr><tr>\n";
61:          }
62:     }
63:     if ($count < $firstDayArray['wday'] || $dayArray['mon'] != $month) {
64:         echo "<td> </td>\n";
65:     } else {
66:         echo "<td>".$dayArray['mday']."    </td>\n";
67:         $start += ADAY;
68:     }
69: }
70: echo "</tr></table>";
71: ?>
72: </body>
73: </html>
```

Din cauză că tabelul va fi indexat pe baza zilelor săptămânii, parcurgem iterativ un vector cu numele acestor zile pe liniile 51–53, afişând fiecare zi în celula proprie pe linia 52. Partea magică a scriptului începe să se desfăşoare în cadrul ultimei bucle for, pornind de la linia 54.

Iniţializăm o variabilă denumită `$count` şi ne asigurăm că bucla se va încheia după 42 de iteraţii. Acest lucru este necesar pentru a fi siguri că avem suficiente celule pentru a afişa informaţiile despre datele calendaristice. În cadrul buclei, transformăm variabila `$start` într-un vector de tip dată calendaristică folosind funcţia `getdate()` şi atribuindu-i rezultatul în vectorul `$dayArray` (linia 55). Din moment ce variabila `$start` reprezintă prima zi a lunii în momentul execuţiei iniţiale a buclei, vom incrementa această valoare cu câte 24 de ore la fiecare iteraţie.

Pe linia 56, verificăm variabila `$count` faţă de numărul 7, folosind operatorul modulus. Deci, blocul de cod din interiorul instrucţiunii `if` va fi executat numai atunci când variabila `$count` este fie zero, fie multiplu de 7. Astfel aflăm dacă trebuie sau nu să încheiem bucla cu totul pentru a trece pe un rând nou.

După ce am stabilit că suntem la prima iteraţie sau la un rând nou, putem să trecem mai departe şi să efectuăm un nou test, pe linia 57. Dacă elementul `mon` (numărul lunii) al vectorului `$dayArray` nu mai este echivalent cu variabila `$month`, înseamnă că am terminat. Reţineţi că vectorul `$dayArray` conţine informaţii despre timestampul `$start`, care este momentul curent din lună pe care îl afişăm. Atunci când `$start` trece dincolo de sfârşitul lunii curente, `$dayArray['mon']` va conţine un număr diferit de valoarea variabilei `$month` care a fost introdusă de utilizator. Testul nostru bazat pe operatorul modulus demonstrează că suntem la sfârşitul unui rând, iar faptul că ne aflăm într-o altă lună înseamnă că trebuie să părăsim cu totul bucla. Presupunând, totuşi, că suntem încă în interiorul lunii pe care vrem să o afişăm, închidem rândul şi începem unul nou pe linia 60.

În cadrul următoarei instrucţiuni `if`, pe linia 63, determinăm dacă trebuie să scriem informaţii despre o dată într-o celulă. Nu toate lunile încep cu o duminică,

aşa că este foarte probabil că vom începe cu câteva celule goale. Doar câteva luni se termină chiar la sfârşitul unuia dintre rândurile noastre, aşa că este, de asemenea, probabil că va trebui să scriem câteva celule goale şi înainte să închidem tabelul HTML. Am stocat informaţiile despre prima zi a lunii în vectorul `$firstDayArray`; în special, putem accesa numărul zile săptămânii stocat în `$firstDayArray['wday']`. Dacă variabila `$count` este mai mică decât acest număr, ştim că nu am ajuns încă la celula corectă în care trebuie să scriem. Prin acelaşi raţionament, dacă variabila `$month` nu mai este egală cu `$dayArray['mon']`, ştim că am ajuns la sfârşitul lunii (dar nu şi la sfârşitul rândului, aşa cum am determinat în cadrul testului bazat pe operatorul modulus efectuat anterior). În ambele cazuri, trebuie să scriem o celulă goală în browser, lucru pe care îl şi facem pe linia 64.

Ultima clauză `else` de pe linia 65 ne permite să facem lucrul cel mai interesant. Deja am stabilit că suntem în interiorul lunii pe care vrem să o afişăm, precum şi că numărul zilei curente, stocat în `$firstDayArray['wday']` se potriveşte cu coloana curentă. Acum trebuie să folosim vectorul asociativ `$dayArray` pe care l-am creat mai devreme în cadrul buclei pentru a scrie ziua lunii şi un spaţiu alb în celulă.

La final, pe linia 67, trebuie să incrementăm variabila `$start`, care conţine *trimestamp*-ul nostru. Pur şi simplu îi adăugăm numărul de secunde dintr-o zi (am definit această valoare pe linia 2) şi suntem gata să începem din nou bucla, cu o nouă valoare de testat în variabila `$start`.

Dacă salvaţi acest listing sub numele de `listing22.3.php`, în amplasaţi în directorul rădăcină al serverului dumneavoastră Web şi îl accesaţi în browser, ar trebui să vedeţi ceva asemănător figurii 22.2 (luna şi anul ar putea fi diferite la dumneavoastră).

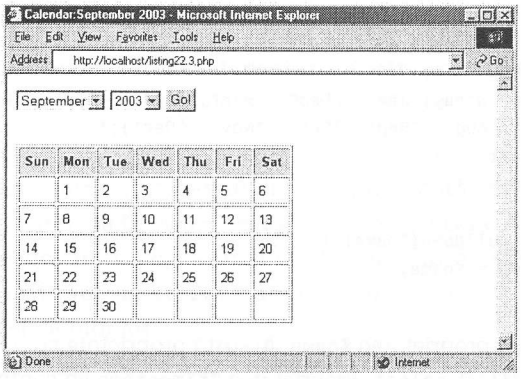


FIGURA 22.2. Formularul şi scriptul aplicaţiei Calendar.

Crearea bibliotecii Calendar

Din cauză că datele calendaristice apar peste tot în interfeţele Web şi pentru că lucrul cu aceste date este, de obicei, relativ complicat, haideţi să încercăm să creăm o bibliotecă de clase care să automatizeze câteva dintre operaţiunile care ar putea fi efectuate cu datele calendaristice. Pe parcursul acestui demers, vom revizui câteva dintre tehnicile pe care le-am discutat deja.

Biblioteca simplă `date_pulldown` pe care o creăm aici va consta din trei elemente `select` diferite: unul pentru ziua lunii, altul pentru lună şi altul pentru an. Atunci când utilizator transmite o pagină, scriptul verifică datele introduse în formular. Dacă există o problemă în ceea ce priveşte datele introduse de utilizator, vom reîncărca pagina lăsând datele introduse în formular. Acest lucru este uşor de realizat cu ajutorul casetelor de text, dar este un pic mai complicat în cazul listelor derulante. Paginile care afişează date calendaristice extrase dintr-o bază de date ne pun o problemă similară. Informaţiile pot, în general, să fie introduse direct în atributele `value` ale elementelor HTML de tip `input`. Datele calendaristice, însă, vor trebui, mai întâi, să fie desfăşurate în valori separate pentru lună, zi şi an, iar după aceea elementele `option` corespunzătoare pot fi selectate.

Clasa `date_pulldown` are drept scop crearea unor liste derulante cu date calendaristice care să fie „lipicioase" (adică să păstreze valorile selectate de la o pagină la alta) şi care să fie uşor de modificat. Pentru a crea clasa noastră, trebuie mai întâi să o declarăm şi să îi creăm un constructor.

Apropo

> Un *constructor* este o funcţie care există în cadrul unei clase şi care este apelată automat atunci când este creată o nouă instanţă a acelei clase.

Putem, de asemenea, să declarăm câteva dintre proprietăţile clasei. Următorul fragment de cod arată începutul clasei noastre:

```
1: class date_pulldown {
2:     var $name;
3:     var $timestamp = -1;
4:     var $months = array("Jan", "Feb", "Mar", "Apr", "May", "Jun",
5:           "Jul", "Aug", "Sep", "Oct", "Nov", "Dec");
6:     var $yearstart = -1;
7:     var $yearend = -1;
8:
9:   function date_pulldown($name) {
10:      $this->name = $name;
11:  }
```

Mai întâi declarăm proprietatea `$name`. Această proprietate va fi folosită pentru a denumi elementele HTML `select`. Proprietatea `$timestamp` va conţine un *timestamp* de tip Unix. Proprietatea `$months` este un vector care va conţine şirurile pe care le vom afişa în listele noastre derulante corespunzătoare lunilor. Proprietăţile `$yearstart` şi `$yearend` sunt, amândouă, create cu valoarea –1, în aşteptarea iniţializării.

Aceste proprietăţi vor conţine, până la urmă, anul de început şi cel de sfârşit al intervalului care va fi prezentat în lista derulantă corespunzătoare anilor.

Constructorul este foarte simplu. Acesta acceptă un şir de caractere, pe care îl atribuie proprietăţii `$name`. Acum că am pus la punct elementele de bază ale clasei noastre, trebuie să creăm un set de metode cu ajutorul cărora codul client să poată iniţializa data. Fragmentul de cod continuă astfel:

```
12:    function setDate_global( ) {
13:        if (!$this->setDate_array($GLOBALS[$this->name])) {
14:            return $this->setDate_timestamp(time());
15:        }
16:
17:        return true;
18:    }
19:
20:    function setDate_timestamp($time) {
21:        $this->timestamp = $time;
22:        return true;
23:    }
24:
25:    function setDate_array($inputdate) {
26:        if (is_array($inputdate) &&
27:            isset($inputdate['mon']) &&
28:            isset($inputdate['mday']) &&
29:            isset($inputdate['year'])) {
30:
31:            $this->timestamp = mktime(11, 59, 59,
32:                $inputdate['mon'], $inputdate['mday'], $inputdate['year']);
33:
34:            return true;
35:        }
36:
37:        return false;
38:    }
```

Dintre aceste metode, `setDate_timestamp()` este cea mai simplă. Ea necesită drept argument un *timestamp* de tip Unix, pe care îl atribuie proprietăţii `$timestamp`.

Metoda `setDate_array()` aşteaptă drept argument un vector asociativ care să conţină cel puţin trei chei: `'mon'`, `'mday'` şi `'year'`. Aceste câmpuri vor conţine datele calendaristice în acelaşi format ca şi rezultatul funcţiei `getdate()`. Acest lucru înseamnă că metoda `setDate_array()` va accepta un vector construit de mână precum cel de mai jos:

```
array('mday'=> 20, 'mon'=>9, 'year' => 2003);
```

dar şi rezultatul unui apel către funcţia `getdate()`:

```
getdate(1032787181);
```

Nu este nici o întâmplare că listele derulante pe care le vom construi mai târziu vor fi produce vectori ce vor conţine câmpurile `'mon'`, `'mday'` şi `'year'`. Metoda foloseşte funcţia `mktime()` pentru a construi un timestamp care, apoi, este atribuit variabilei `$timestamp`.

Metoda `setDate_global()` este apelată implicit. Aceasta încearcă să găsească o variabilă globală cu acelaşi nume ca al obiectului — adică numele conţinut de proprietatea `$name`. Această variabilă este pasată metodei `setDate_array()`. Dacă există o variabilă globală cu structura corectă, metoda `setDate_array()` o foloseşte pentru a crea variabila `$timestamp`. În caz contrar, este folosită data şi ora curentă.

Intervalele pentru zile şi luni sunt fixate, dar anii reprezintă o problemă din acest punct de vedere. Vom crea câteva metode care să îi permită codului client să iniţializeze propriul interval de ani (deşi, pe de altă parte, punem la dispoziţie şi astfel de interval implicit).

```
39:    function setYearStart($year) {
40:        $this->yearstart = $year;
41:    }
42:
43:    function setYearEnd($year) {
44:        $this->yearend = $year;
45:    }
46:
47:    function getYearStart() {
48:        if ($this->yearstart < 0) {
49:            $nowarray = getdate(time());
50:            $this->yearstart = $nowarray[year]-5;
51:        }
52:
53:        return $this->yearstart;
54:    }
55:
56:    function getYearEnd() {
57:        if ($this->yearend < 0) {
58:            $nowarray = getdate(time());
59:            $this->yearend = $nowarray[year]+5;
60:        }
61:        return $this->yearend;
62:    }
```

Metodele `setYearStart()` şi `setYearEnd()` sunt cât se poate de clare. Un an este atribuit direct proprietăţii corespunzătoare. Metoda `getYearStart()` verifică dacă proprietatea `$yearstart` este iniţializată corect. Dacă această proprietate nu a fost iniţializată, `getYearStart()` îi atribuie o valoarea mai mare cu cinci ani decât anul curent. Metoda `getYearStart()` efectuează o operaţiune similară. Acum suntem gata să creăm partea lucrativă a clasei:

```
63:    function output() {
64:        if ($this->timestamp < 0) {
65:            $this->setDate_global();
66:        }
```

```
67:         $datearray = getdate($this->timestamp);
68:         $out  = $this->day_select($this->name, $datearray);
69:         $out .= $this->month_select($this->name, $datearray);
70:         $out .= $this->year_select($this->name, $datearray);
71:         return $out;
72:     }
73:
74:     function day_select($fieldname, $datearray)  {
75:         $out = "<select name=\"$fieldname"."[mday]\">\n";
76:         for ($x=1; $x<=31; $x++) {
77:             $out .= "<option value=\"$x\"".($datearray['mday']==($x)
78:                 ?" SELECTED":"").">".sprintf("%02d", $x ) ."\n";
79:         }
80:         $out .= "</select>\n";
81:         return $out;
82:     }
83:
84:     function month_select($fieldname, $datearray) {
85:         $out = "<select name=\"$fieldname"."[mon]\">\n";
86:         for ($x = 1; $x <= 12; $x++) {
87:             $out .= "<option value=\"".($x)."\"".($datearray['mon']==($x)
88:                 ?" SELECTED":"").">".$this->months[$x-1]."\n";
89:         }
90:         $out .= "</select>\n";
91:         return $out;
92:     }
93:
94:     function year_select($fieldname, $datearray) {
95:         $out = "<select name=\"$fieldname"."[year]\">";
96:         $start = $this->getYearStart();
97:         $end = $this->getYearEnd();
98:         for ($x= $start; $x < $end; $x++) {
99:             $out .= "<option value=\"$x\"".($datearray['year']==($x)
100:                 ?" SELECTED":"").">$x\n";
101:         }
102:         $out .= "</select>\n";
103:         return $out;
104:     }
105: }
```

Metoda `output()` coordonează execuţia celei mai mari părţi a acestei porţiuni de cod. Mai întâi, ea verifică proprietatea `$timestamp`. În afara cazului în care utilizatorul a apelat una dintre metodele `setDate`, valoarea lui `$timestamp` va fi, în acest moment, –1 aşa că va fi apelată automat metoda `setDate_global()`. *Timestamp*-ul va fi pasat funcţiei `getdate()` în vederea construirii unui vector conţinând elementele datei calendaristice şi, după aceasta, este apelată o metodă pentru producerea fiecăruia dintre listele derulante.

Metoda `day_select()` nu face altceva decât să construiască un element HTML `select` care conţine un element `option` pentru fiecare dintre cele 31 de zile posibile ale unei luni. Data curentă a obiectului este stocată în variabila argument `$datearray`, care este folosită în timpul construirii elementului pentru a defini drept atribut selectat opţiunea corespunzătoare datei cu care se lucrează.

Funcţia `sprintf()` formatează număr zilei, adăugând un zero la începutul zilelor cu numărul de la 1 la 9. Metodele `month_select()` şi `year_select()` folosesc un raţionament similar pentru a construi listele derulante corespunzătoare lunii şi, respectiv, anului.

De ce am fragmentat codul pentru elaborarea datelor de ieşire în patru metode, în loc să creăm pur şi simplu un singur bloc de cod? Atunci când construim o clasă, trebuie să avem în minte două tipuri de utilizatori: programatorul client care va dori să instanţieze un obiect din clasa `date_pulldown` şi programatorul client care va dori să derive clase copil de la clasa `date_pulldown`, pentru a-i rafina funcţionalitatea. Pentru cel dintâi, vrem să punem la dispoziţie o interfaţă clară prin care să poată fi accesată funcţionalitatea clasei. Programatorul poate să instanţieze obiectul, să îi iniţializeze data şi să apeleze metoda `output()`. Pentru cel din urmă, vrem să facilităm modificarea diverselor elemente ale funcţionalităţii clasei. Fragmentând codul astfel în metode discrete, permitem derivarea de clase copil care să modifice aspecte limitate ale funcţionalităţii fără să deranjeze în vreun fel întregul. Dacă o clasă copil are nevoie să reprezinte lista derulantă corespunzătoare anului sub forma a două butoane radio, de pildă, tot ce trebuie să facă programatorul este să redefinească metoda `year_select()`.

Listingul 22.4 conţine un fragment de cod care apelează biblioteca de clase. Înainte să încercaţi să executaţi acest cod, luaţi toate fragmentele codului clasei pe care le-am discutat mai sus, puneţi-le într-un fişier denumit `date_pulldown.class.php` şi plasaţi acest fişier în directorul rădăcină al serverului dumneavoastră Web. În câteva momente veţi apela acest fişier, aşa că ar fi bine să se găsească acolo unde trebuie!

LISTINGUL 22.4. Folosirea clasei `date_pulldown`.

```
<html>
<head>
<title>Listing 22.4 Using the date_pulldown Class</title>
</head>
<?php
include("date_pulldown.class.php");
$date1 = new date_pulldown("fromdate");
$date2 = new date_pulldown("todate");
$date2 = new date_pulldown("foundingdate");
 $date3->setYearStart(1972);
if (empty($foundingdate)) {
   $date3->setDate_array(array('mday'=>26, 'mon'=>4, 'year'=>1984));
}
?>
<body>
<form>
<P><strong>From:</strong><br>
<?php echo $date1->output(); ?>
<P><strong>To:</strong><br>
<?php echo $date2->output(); ?>
```

LISTINGUL 22.4. (*continuare*)

```
21: <P><strong>Company Founded:</strong><br>
22: <?php echo $date3->output(); ?>
23: <P><input type="submit" name="submit" value="submit"></p>
24: </form>
25: </body>
26: </html>
```

Includem fişierul `date_pulldown.class.php` pe linia 6 şi, din moment ce am inclus acest fişier cu clase, putem să folosim oricare dintre metodele sale. Folosim comportamentul implicit al clasei pentru toate listele derulante, cu excepţia instanţei numite `"foundingdate"`. Pentru acest obiect, redefinim anul de start implicit, iniţializându-l la valoarea 1972, pe linia 10. Pe linia 12, atribuim acestei liste derulante o dată calendaristică arbitrară ce va fi afişată până ce formularul este transmis.

Apropo

Acesta este doar porţiunea de interfaţă a formularului, fără nici o acţiune şi fără nici o metodă; trebuie să creaţi propria acţiune sau metodă pentru ca acest cod să facă efectiv ceva!

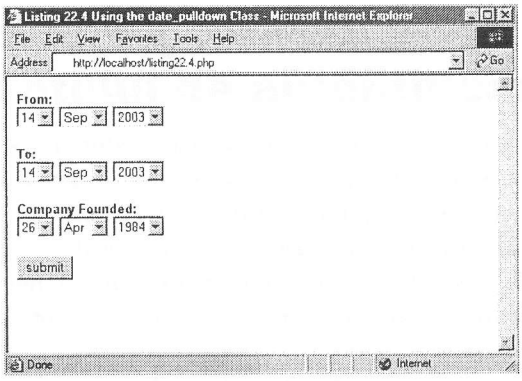


FIGURA 22.3. Listele derulante generate de către clasa `date_pulldown`.

Rezumat

În acest capitol, aţi folosit funcţiile PHP de operare cu datele calendaristice pe care le-aţi învăţat mai devreme în această carte pentru a construi o aplicaţie de tip calendar. Aţi învăţat cum să verificaţi validitatea datelor introduse de utilizator folosind funcţia `checkdate()`. Aţi parcurs pas cu pas un script ilustrativ, care aplică o parte dintre instrumentele pe care le-aţi învăţat anterior şi aţi construit o bibliotecă de clase care automatizează câteva dintre cele mai neplăcute aspecte ale lucrului cu datele calendaristice în cadrul formularelor HTML.

Întrebări şi răspunsuri

Î. Există funcţii care să convertească datele între diverse calendare?

R. Da. PHP ne pune la dispoziţie o serie întreagă de funcţii care operează cu calendare alternative. Puteţi să citiţi despre acestea în Manualul oficial de PHP la adresa `http://www.php.net/manual/ro/ref.calendar.php`.

Atelier

Atelierul este conceput să anticipeze posibile întrebări, să vă ajute să recapitulaţi cele învăţate şi să începeţi să vă puneţi cunoştinţele în practică.

Test

1. Ce funcţie PHP am folosit pentru a verifica validitatea unei date calendaristice?
2. Ce funcţie PHP am folosit pentru a crea un *timestamp*?
3. Ce funcţie PHP am folosit pentru a crea un vector asociativ conţinând informaţii despre o dată calendaristică?

Soluţii

1. `checkdate()`
2. `mktime()`
3. `getdate()`

Activitate

Folosiţi noua şi nemaipomenita clasă cu liste derulante în cadrul unui formular propriu. Creaţi un script care să preia datele selectate şi să afişeze selecţia efectuată de utilizator.

CAPITOLUL 23
Restricționarea accesului la aplicațiile dumneavoastră

Acest capitol explică metodele de folosire a serverului Apache în vederea restricţionării accesului la anumite părţi ale unui site Web pe baza identităţii unui utilizator sau a unor informaţii despre solicitările acestuia. De partea aplicaţiei, puteţi să vă creaţi propriul mecanism de validare a utilizatorilor şi să verificaţi identitatea acestora prin intermediul cookie-urilor.

În acest capitol, veţi învăţa cum să:

- restricţionaţi accesul pe baza utilizatorului, a adresei sale de IP, a numelui de domeniu şi a versiunii browserului;
- folosiţi instrumentele de gestiune a utilizatorilor puse la dispoziţie împreună cu Apache;
- stocaţi informaţii în cookie-uri şi să extrageţi aceste informaţii;
- folosiţi cookie-urile pentru autentificare.

Prezentare generală a conceptului de autentificare

Autorizarea şi autentificarea sunt necesităţi uzuale pentru multe site-uri Web. *Autentificarea* stabileşte identitatea părţilor care comunică. Puteţi să vă autentificaţi folosind ceea ce ştiţi (o parolă, un cookie), ceea ce aveţi (un card de identificare, o cheie), sau ceea ce sunteţi (amprenta dumneavoastră, modelul vaselor retiniene), sau o combinaţie a acestor elemente. În contextul Web-ului, autentificarea foloseşte, de cele mai multe ori, doar parolele şi certificatele.

Autorizarea se referă la protecţia accesului la resurse. Puteţi să oferiţi autorizare pe baza mai multor factori, cum ar fi adresa de IP de la care se conectează utilizatorul, browserul folosit de utilizator, conţinutul pe care încearcă să îl acceseze utilizatorul, sau identitatea utilizatorului (determinată anterior, în urma procesului de autentificare).

Apache include câteva module care oferă posibilitatea autentificării şi a controlării accesului şi care pot fi folosite atât la protejarea datelor dinamice cât şi la protejarea datelor statice. Puteţi folosi oricare dintre aceste module sau puteţi să implementaţi propriul sistem de control al accesului la nivel de aplicaţie şi să puneţi la dispoziţia utilizatorului ecrane de autentificare, facilităţi de autentificare unică, precum şi alte funcţii avansate.

Autentificarea programului client

Utilizatorii pot fi autentificaţi atât în scopul autorizării cât şi în scopul urmăririi. Specificaţia de HTTP ne pune la dispoziţie două mecanisme de autentificare simplă (basic) şi „digest". În ambele cazuri, procedura este următoarea:

1. Un program client (un browser, de exemplu) încearcă să acceseze date restricţionate de pe un server Web.
2. Apache verifică dacă respectivul program client a transmis un nume de utilizator şi o parolă. Dacă acest lucru nu s-a întâmplat, Apache returnează un cod de stare `HTTP 401`, care indică faptul că este necesară autentificarea utilizatorului.
3. Programul client citeşte răspunsul şi îi solicită utilizatorului numele de utilizator şi parola necesare (de obicei prin intermediul unei casete de dialog).
4. Programul client încearcă din nou să acceseze pagina Web, de data aceasta transmiţând numele de utilizator şi parola în cadrul solicitării HTTP. Programul client păstrează în memorie numele de utilizator şi parola şi le transmite în cadrul tuturor solicitărilor ulterioare către acelaşi site, astfel încât utilizatorul să nu fie nevoit să le tasteze din nou pentru fiecare solicitare (de fiecare dată când dă clic pe un link, de exemplu).
5. Apache verifică validitatea acreditării şi, în funcţie de aceasta, permite sau refuză accesul pe baza identităţii utilizatorului şi a celorlalte reguli de acces.

În cadrul mecanismului de autentificare simplă, numele de utilizator şi parola sunt transmise în clar, ca parte a anteturilor solicitării HTTP. Acest lucru reprezintă un risc de securitate deoarece un atacator ar putea să tragă cu urechea foarte uşor la conversaţiile dinte server şi browser, aflând astfel numele de utilizator şi parola şi folosindu-le, apoi, după plac.

Autentificare de tip „digest" oferă un nivel mai înalt de securitate deoarece, în acest caz, în locul parolei în clar este trimisă o versiune codificată. Digest-ul este bazat pe o combinaţie dintre mai mulţi parametri, printre care şi numele de utilizator, parola şi metoda de solicitare folosită. Serverul poate, apoi, să calculeze şi el digest-ul şi să verifice dacă clientul ştie într-adevăr parola, chiar dacă aceasta nu a fost transmisă prin reţea.

Apropo

> Un *algoritm de digest* este o operaţiune matematică ce preia un text şi returnează un alt text, un digest (rezumat) sau hash, care identifică în mod unic textul original. Un algoritm bun ar trebui să se asigure că, din motive practice, texte originale diferite produc hash-uri diferite, iar că textul original nu poate fi dedus din hash. MD5 este numele celui mai frecvent folosit algoritm de hash.

Din păcate, în ciuda faptului că specificaţia este disponibilă de multă vreme, numai browserele foarte noi suportă autentificarea pe bază de digest. Acest lucru înseamnă că, din motive pragmatice, autentificarea pe bază de digest trebuie rezervată cazurilor în care aveţi control asupra browserului folosit de către utilizatori, cum ar fi, de exemplu, cazul intranetului unei companii.

În orice caz, pentru ambele tipuri de autentificare, informaţia solicitată este transmisă în mod neprotejat prin reţea. O alegere mai bună ar fi să securizaţi serverul dumneavoastră folosind protocolul HTTP prin intermediul protocolului SSL, aşa cum este descris în capitolul 27, intitulat *Configurarea unui server Web securizat*.

Metode de gestiune a utilizatorilor

Atunci când modulul de autentificare primeşte numele de utilizator şi parola din partea programului client, acesta trebuie să verifice dacă ele sunt valide pe baza unei liste existente de utilizatori. Numele de utilizatori şi parolele pot fi stocate într-o varietate de moduri. Apache poate să folosească atât mecanisme de autentificare bazate pe fişiere cât şi mecanisme de autentificare ce folosesc baze de date. Modulele furnizate de terţi pot să ne pună la dispoziţie mecanisme suplimentare cum ar fi Lightweight Directory Access Protocol (LDAP) şi Network Information Services (NIS).

Funcționalitatea modulului de autentificare al serverului Apache

Apache ne pune la dispoziţie un cadru de bază, precum şi directivele necesare pentru autentificare şi pentru controlul accesului. Modulele de autentificare oferă sprijin pentru validarea parolelor pe baza unui mecanism specific. Utilizatorii pot fi, opţionali, organizaţi în grupuri, ceea ce uşurează procesul de gestiune a regulilor de control.

Apache are trei directive predefinite ce ţin de autentificare şi care sunt folosite de oricare dintre modulele de autentificare: `AuthName`, `AuthType` şi `Require`.

`AuthName` acceptă un argument de tip şir de caractere, numele de autentificare pentru respectivul realm (incintă). Un realm este o zonă logică a serverului Web pentru care solicitaţi parola. Acesta va fi afişat în caseta de dialog a browserului.

`AuthType` specifică tipul de autentificare pe care trebuie să o efectueze browserul: de bază sau digest.

`Require` vă permite să specificaţi o listă de utilizatori sau de grupuri cărora li se va permite accesul. Sintaxa este `Require user` urmat de unul sau mai multe nume de utilizator, sau `Require group` urmat de unul sau mai multe nume de grupuri. De exemplu:

```
Require user joe bob
```

sau

```
Require group employee contractor
```

Dacă vreţi să permiteţi accesul tuturor celor care furnizează un nume de utilizator valid şi o parolă, puteţi face acest lucru astfel:

```
Require valid-user
```

Folosind directivele de mai sus, puteţi să controlaţi cine are acces la anumite host-uri virtuale, directoare, fişiere şi aşa mai departe. Deşi autentificarea şi autorizarea sunt concepte separate, în practică, în cazul serverului Apache, ele se petrec simultan. Accesul este permis pe baza unei anumite identităţi a utilizatorului sau a apartenenţei acestuia la un anumit grup. Anumite module furnizate de terţi, cum ar fi modulele bazate pentru LDAP, permit o separare mai clară între autentificare şi autorizare.

Modulele de autentificare incluse în pachetul Apache oferă:

- **stocare pe server** — este permisă folosirea fişierelor text sau a bazelor de date care să conţină nume de utilizatori şi informaţii despre grupurile de utilizatori;
- **gestiunea utilizatorilor** — sunt puse la dispoziţie instrumente pentru crearea şi gestiunea utilizatorilor şi a grupurilor stocate pe server;
- **informaţii despre autorizare** — este posibilă specificarea faptului că rezultatele modulului sunt sau nu relevante din punct de vedere al autorizării.

Apropo

Uneori, utilizatorilor li se refuză accesul pentru că informaţiile lor nu sunt găsite în baza de date folosită de către modul, sau pentru că nici o regulă de autentificare nu se potriveşte cu informaţiile lor. În acest caz, va avea loc una dintre următoarele situaţii:

- dacă modulul are specificaţia conform căreia rezultatele sale sunt relevante din punct de vedere al autorizării, utilizatorului i se va refuza accesul, iar Apache va returna o eroare.
- dacă modulul are specificaţia conform căreia rezultatele sale nu sunt relevante din punct de vedere al autentificării, alte module au posibilitatea să autentifice utilizatorul.

Acest sistem vă permite să aveţi un modul principal de autorizare care să ştie datele majorităţii utilizatorilor şi să puteţi avea, suplimentar, alte module care să autentifice restul utilizatorilor.

Autentificarea prin folosirea fişierelor

Modului Apache `mod_auth` pune la dispoziţie autentificare de bază care foloseşte fişiere text conţinând nume de utilizatori şi parole, similare cu cele folosite de sistemul de autentificare Unix şi anume fişierele `etc/passwd` şi `etc/groups`.

Stocarea pe server

Trebuie să specificaţi fişierul care conţine lista numelor de utilizatori şi parolele şi, opţional, fişierul care conţine lista grupurilor.

Fişierul cu utilizatori este un fişier de parole stil Unix, care conţine numele utilizatorilor şi parolele criptate. Intrările arată ca cea de mai jos, sub Unix, folosind un algoritmul `crypt`:

```
admin:iFrlxqgØQ6RQ6
```

iar în Windows, folosind algoritmul MD5:

```
admin:$apr1$Ug3.....$jVTedbQWBKTfXsn5jK6UX/
```

Fişierul cu grupuri conţine o listă de grupuri şi numele utilizatorilor care aparţin fiecărui grup, separate prin spaţii, la fel ca în exemplul de mai jos:

```
web: admin joe Daniel
```

Directivele `AuthUserFile` şi `AuthGroupFile` primesc un argument constând din calea către fişierul cu utilizatori şi calea către fişierul cu grupuri. Fişierul cu grupuri este opţional.

Gestiunea utilizatorilor

Apache include utilitarul `htpasswd` pentru Unix, şi utilitarul `htpasswd.exe` pentru Windows; acestea sunt concepute pentru a vă ajuta să gestionaţi fişierele cu parolele utilizatorilor. Ambele versiuni funcţionează la fel, dar versiunea pentru Windows foloseşte un alt algoritm de criptare a parolei. Criptarea este transparentă pentru utilizator şi administrator. Prima oară când adăugaţi un utilizator, trebuie să tastaţi:

```
/usr/local/apache2/bin/htpasswd -c fisier IDUtilizator
```

unde `fisier` este fişierul cu parole care va conţine lista numelor de utilizatori şi parolele, iar IDUtilizator este numele de utilizator pe care doriţi să îl adăugaţi. Vi se va cere să introduceţi o parolă, iar fişierul va fi creat. De exemplu, sub Linux/Unix, linia:

```
/usr/local/apache2/bin/htpasswd -c /usr/local/apache2/conf/htusers admin
```

va crea un fişier cu parole în calea `/usr/local/apache2/conf/htusers` şi va adăuga în acest fişier utilizatorul `admin`.

Acelaşi lucru este disponibil şi sub sistemul de operare Windows, unde operaţia din linie de comandă ar arăta ca în exemplul de mai jos:

```
htpasswd -c "C:\Program Files\Apache Group\Apache2\conf\htusers" admin
```

Opţiunea –c din linia de comandă îi spune utilitarului `htpasswd` că trebuie să creeze fişierul. Atunci când vreţi să adăugaţi utilizatori într-un fişier care există deja, nu folosiţi opţiunea –c, altminteri fişierul va fi suprascris.

Este important să stocaţi fişierul cu parole în altundeva decât în directorul rădăcină de documente, pentru ca acesta să nu fie accesibil prin intermediul browserului. În caz contrar, un atacator ar putea să descarce fişierul şi să obţină o listă cu numele de utilizatori şi parolele dumneavoastră. Deşi parolele sunt criptate, atunci când ai acest fişier, este mult mai uşor să efectuezi un atac de tip brute force (forţă brută) pentru a încerca să ghiceşti parolele.

Autorizarea

Directiva `AuthAuthoritative` preia o valoare de tip `on` sau `off`. Valoarea inițială este `on`, ceea ce înseamnă că rezultatele de autentificare ale modulului au relevanță pentru autorizare. Acest lucru înseamnă, mai departe, că dacă utilizatorul nu este găsit sau nu se potrivește cu regulile de acces, accesul îi va fi refuzat.

Folosirea modulului `mod_auth`

Listingul 23.1 arată un model de configurare, care restricționează accesul la directorul `private` din rădăcina de documente, astfel încât doar utilizatorii autentificați, care se află în fișierul cu parole `htusers`, pot să îl acceseze. Observați că nu este prezentă directiva opțională `AuthGroupFile`.

LISTINGUL 23.1. Exemplu de autentificare folosind fișiere.

```
1: <Directory /usr/local/apache2/htdocs/private>
2: AuthType Basic
3: AuthName "Private Area"
4: AuthUserFile /usr/local/apache2/conf/htusers
5: AuthAuthoritative on
6: Require valid-user
7: </Directory>
```

Controlul accesului folosind fișiere tip bază de date

Stocarea numelor de utilizatori și a parolelor în fișiere text este facilă, dar nu reprezintă o soluție scalabilă. Apache trebuie să deschidă și să citească fișierul secvențial pentru a căuta un anumit utilizator. Atunci când numărul de utilizatori crește, această operațiune începe să consume foarte mult timp. Modulul `mod_auth_dbm` vă permite să înlocuiți fișierele text cu fișiere de tip bază de date indexate, care pot suporta un număr mult mai mare de utilizatori fără degradarea semnificativă a performanțelor. Activarea acestui modul are loc atunci când compilați sursele de Apache, folosind opțiunea de configurare `--enable-module=dbm`.

Stocarea pe server

Modulul `mod_auth_dbm` pune la dispoziție două directive, `AuthDBMFile` și `AuthDBMGroupFile`, care indică fișierele tip bază de date conținând numele de utilizatori și parolele și, respectiv, grupurile de utilizatori. Spre deosebire de directivele pentru fișiere text, ambele aceste directive pot să indice același fișier care va combina atât utilizatorii cât și parolele.

Gestiunea utilizatorilor

Apache pune la dispoziție un script scris în Perl (`dbmmanage` pentru Unix și `dbmmanage.pl` pentru Windows) care vă permite să creați și să gestionați utilizatorii și grupurile stocate în fișierul tip bază de date.

Sub Unix, s-ar putea să doriți să editați prima linie a scriptului astfel încât aceasta să găsească interpetorul Perl instalat pe sistemul dumneavoastră. Pe sistemele Windows trebuie să instalați pachetul suplimentar MD5. Dacă folosiți versiunea ActiveState de Perl, rulați utilitarul de gestiune a pachetelor Perl și tastați:

```
install Crypt-PasswdMD5
```

Pentru a adăuga un utilizator în baza de date sub Unix, tastați:

```
dbmmanage dbfile adduser userid
```

Sub Windows, tastați:

```
perl dbmmanage.pl dbfile adduser userid
```

Vi se va solicita introducerea unei parole, iar utilizatorul va fi adăugat în fișierul de tip bază de date existent sau va fi creat un nou fișier în cazul în care acesta nu există.

Atunci când adăugați un utilizator, puteți, opțional, să specificați grupurile din care acesta trebuie să facă parte, sub forma unor argumente separate prin virgule. Următoarea comandă adaugă utilizatorul `daniel` în fișierul bază de date din calea `/usr/local/apache2/conf/dbmusers` și îl înscrie în grupurile `employee` și `engineering`:

```
dbmmanage /usr/local/apache2/conf/dbmusers adduser daniel employee,engineering
```

Dacă aveți vreodată nevoie să ștergeți utilizatorul `daniel`, puteți folosi următoarea comandă:

```
dbmmanage dbfile delete daniel
```

Programul `dbmmanage` admite și alte opțiuni suplimentare. Puteți găsi lista completă a informațiilor despre sintaxa sa în pagina de manual dedicată lui `dbmmanage` sau invocând `dbmmanage` fără nici un alt argument.

Apropo

Apache 2.0 pune la dispoziție și un alt utilitar, `htdbm`, care nu depinde de Perl și oferă exact aceeași funcționalitate ca și `dbmmanage`.

Folosirea serverului Apache pentru controlul accesului

Modulul `mod_access`, activat de la instalare, vă permite să restricționați accesul la resurse pe baza unor parametri ai solicitării transmise de către programul client, cum ar fi prezența sau absența unui anumit antet, adresa de IP sau numele host-ului clientului.

Implementarea regulilor de acces

Puteţi specifica reguli de acces folosind directivele `Allow` (permite) şi `Deny` (refuză). Fiecare dintre aceste directive preia o listă de argumente cum ar fi adresele de IP, variabilele mediului de operare şi numele de domeniu.

Accesul de tip `Allow/Deny` pe baza adresei de IP

Puteţi să refuzaţi sau să permiteţi accesul unui client pe baza adresei sale de IP:

```
Allow from 10.0.0.1 10.0.0.2 10.0.0.3
```

Puteţi, de asemenea, să specificaţi intervale de adrese IP sau o pereche reţea/mască. În plus, mai puteţi specifica primul, primii doi, sau primii trei octeţi ai unei adrese de IP. Orice adresă de IP care conţine aceşti octeţi va declanşa regula respectivă. De exemplu, regula:

```
Deny from 10.0
```

va fi declanşată de adresele care încep cu 10.0, cum ar fi 10.0.1.0 sau 10.0.0.1.

Puteţi, de asemenea, să folosiţi adresa de IP şi masca de subreţea; adresa de IP specifică reţeaua, iar masca de subreţea specifică ce octeţi fac parte din prefixul de reţea şi care dintre aceştia aparţin nodurilor. Regula:

```
Allow from 10.0.0.0/255.255.255.0
```

va fi declanşată de adresele de IP 10.0.0.1, 10.0.0.2 şi aşa mai departe până la 10.0.0.254.

Mai puteţi şi să specificaţi masca de subreţea prin intermediul octeţilor celor mai semnificativi. De exemplu, aţi putea rescrie regula de mai sus sub forma:

```
Allow from 10.0.0.0/24
```

Accesul de tip `Allow/Deny` pe baza numelui de domeniu

Puteţi controla accesul pe baza unor anumite nume de host, sau pe baza unor nume parţiale de domeniu. De exemplu, `Allow from example.com` se va declanşa şi pentru `www.example.com` şi pentru `foo.example.com` şi aşa mai departe.

Apropo

Activarea accesului pe bază de nume de domeniu îl obligă pe serverul Apache să efectueze o interogare DNS inversată asupra adresei clientului, indiferent de valoarea directivei `HostNameLookups`. Acest lucru are un impact asupra performanţei.

Accesul de tip `Allow/Deny` pe baza variabilelor mediului de operare

Puteți specifica reguli de acces pe baza prezenței anumitor variabile ale mediului de operare, prefixând numele variabilei cu `env=`. Puteți folosi această variantă atunci când doriți să permiteți sau să refuzați accesul pentru anumite browsere sau versiuni ale browserelor, pentru a împiedica anumite site-uri să direcționeze utilizatorii prin linkuri către resursele dumneavoastră și așa mai departe. Pentru ca acest exemplu să funcționeze așa cum este intenționat, programul client trebuie să transmită antetul `User-Agent`:

De exemplu:

```
BrowserMatch MSIE iexplorer
Deny from env=iexplorer
```

Din cauză că programul client este cel care trimite antetul `User-Agent`, acesta ar putea să omită sau să modifice antetul respectiv, dar majoritatea utilizatorilor nu vor face acest lucru așa că tehnica va funcționa în majoritatea cazurilor.

Accesul de tip `Allow/Deny` pentru toți clienții

Cuvântul cheie `all` indică toate conexiunile clienților. Puteți specifica `Allow from all` sau `Deny from all` pentru a permite sau refuza accesul din partea tuturor clienților.

Evaluarea regulilor de acces

Puteți avea mai multe reguli de acces `Allow` și `Deny`. Puteți alege ordinea în care aceste reguli sunt evaluate folosind directiva `Order`. Regulile care sunt evaluate mai târziu au o prioritate mai mare. `Order` acceptă un singur argument care poate fi `Deny,Allow`, `Allow,Deny` sau `Mutual-Failure`. Argumentul `Deny,Allow` este valoarea implicită a directivei `Order`. Rețineți că, în valoarea argumentelor, nu există spații.

Argumentul `Deny,Allow`

Valoarea `Deny,Allow` indică faptul că directivele `Deny` trebuie evaluate înaintea directivelor `Allow`. Atunci când folosiți `Deny,Allow`, clientului îi este permis implicit accesul dacă nu există nici o directivă `Allow` sau `Deny` sau dacă clientul nu se încadrează în nici una dintre reguli. În cazul în care clientul se încadrează într-o regulă `Deny`, atunci el nu va avea acces decât dacă nu cumva se încadrează și într-una dintre regulile `Allow`, care va avea prioritate față de regula `Deny`, deoarece directivele `Allow` sunt evaluate ultimele și au prioritate mai mare.

Listingul 23.2 arată cum trebuie configurat serverul Apache pentru a permite accesul la directorul `/private` clienților care vin din rețeaua locală sau din domeniul `example.com` și pentru a refuza accesul tuturor celorlalți clienți.

LISTINGUL 23.2. Un exemplu de configurare a controlului accesului de tip `Deny,Allow`.

```
1: <Location /private>
2: Order Deny,Allow
3: Deny from all
4: Allow from 10.0.0.0/255.255.255.0 example.com
5: </Location>
```

Argumentul `Allow,Deny`

`Allow,Deny` indică faptul că directivele `Allow` sunt evaluate înaintea directivelor `Deny`. Atunci când folosim `Allow,Deny`, clientului i se refuză implicit accesul dacă nu există directive `Allow` şi `Deny` sau dacă clientul nu se încadrează în nici una dintre reguli. În cazul în care clientul se încadrează într-o regulă `Allow`, i se va permite accesul numai dacă nu cumva se potriveşte şi cu o regulă `Deny`, care are prioritate.

Observaţi că prezenţa directivei `Order Allow,Deny` fără nici o regulă `Allow` sau `Deny` duce la refuzarea accesului la resursa respectivă pentru toată lumea, deoarece comportamentul implicit al serverului în acest caz este acela de a refuza accesul.

Listingul 23.3 permite accesul oricui, cu excepţia celor care se conectează de la un anumit host.

LISTINGUL 23.3. Un exemplu de configurare a controlului accesului de tip `Allow,Deny`.

```
1: <Location /some/location/>
2: Order Allow,Deny
3: Allow from all
4: Deny from host.example.com
5: </Location>
```

Argumentul `Mutual-Failure`

În cazul folosirii valorii `Mutual-Failure`, clientului i se va permite accesul numai dacă se încadrează într-o regulă `Allow` şi nu se încadrează în nici o directivă `Deny`.

Combinarea metodelor de acces în Apache

În secţiunile precedente, aţi învăţat cum să restricţionaţi accesul pe baza identităţii utilizatorului sau a informaţiilor transmise odată cu solicitarea HTTP. Directiva `Satisfy` vă permite să stabiliţi dacă ambele tipuri de condiţii de acces trebuie satisfăcute în acelaşi timp pentru ca utilizatorului să i se permită accesul. `Satisfy` acceptă un singur parametru, care poate fi `all` sau `any`.

`Satisfy all` înseamnă că nu i se va acorda acces clientului dacă nu furnizează un nume de utilizator şi o parolă validă şi, în acelaşi timp, nu îndeplineşte condiţiile restricţiilor de acces. `Satisfy any` înseamnă că i se va acorda clientului

accesul dacă furnizează un nume de utilizator şi o parolă validă sau dacă îndeplineşte condiţiile restricţiilor de acces.

De ce este utilă această directivă? De exemplu, aţi putea dori să oferiţi acces liber la site-ul dumneavoastră Web utilizatorilor care se conectează prin reţeaua internă, de la adrese de încredere, dar să solicitaţi autentificare cu nume de utilizator şi parolă celor care se conectează prin Internet. Listingul 23.4 demonstrează exact acest lucru.

LISTINGUL 23.4. Combinarea autentificării cu regulile de control al accesului.

```
1: <Location /restricted>
2: Allow from 10.0.0.0/255.255.255.0
3: AuthType Basic
4: AuthName "Intranet"
5: AuthUserFile /usr/local/apache2/conf/htusers
6: AuthAuthoritative on
7: Require valid-user
8: Satisfy any
9: </Location>
```

Apropo

Controlul accesului pe baza conexiunii sau informațiilor pasate odată cu solicitarea HTTP nu este o metodă complet sigură. Deşi oferă un nivel de protecţie adecvat în majoritatea cazurilor, regulile se bazează pe integritatea serverelor dumneavoastră DNS şi pe infrastructura dumneavoastră de reţea. Dacă un atacator dobândeşte controlul asupra serverelor dumneavoastră DNS, sau dacă routerele şi firewall-urile dumneavoastră nu sunt configurate corect, atacatorul poate să modifice cu uşurinţă înregistrările de nume de domeniu autorizate astfel încât să indice adresa maşinii sale sau, respectiv, să pretindă că se conectează de la o adresă de IP autorizată.

Limitarea accesului pe baza metodelor HTTP

În general, veţi dori să folosiţi directive de control al accesului pentru toate tipurile de solicitări HTTP pe care le fac programele client, iar acesta este comportamentul implicit al serverului. În unele cazuri, totuşi, veţi dori ca autentificarea şi regulile de acces să se aplice numai pentru anumite metode HTTP, cum ar fi `GET` sau `HEAD`.

În containerul `<Limit>` pot fi incluse o serie de metode şi directive care se aplică solicitărilor care conţin aceste metode HTTP. Lista completă a metodelor care pot fi folosite este `GET`, `POST`, `PUT`, `DELETE`, `CONNECT`, `OPTIONS`, `TRACE`, `PATCH`, `PROPFIND`, `PROPPATCH`, `MKCOL`, `LOCK` şi `UNLOCK`.

Secţiunea `<LimitExcept>` oferă o funcţionalitate suplimentară, ea conţinând directive care se vor aplica solicitărilor care nu conţin respectivele metode.

Listingul 23.5 arată un exemplu din fişierul original de configurare al serverului Apache. Secţiunile `<Limit>` şi `<LimitExcept>` permit metodele care doar citesc informaţii dar refuză solicitările ce folosesc alte metode care ar putea modifica

conţinutul sistemului de fişiere, aşa cum este PUT. Pentru mai multe informaţii despre pleiada de opţiuni disponibile în acest context, consultaţi documentaţia Apache de la adresa `http://httpd.apache.org/docs-2.0/mod/core.html`.

LISTINGUL 23.5. Restricţionarea accesului pe baza regulilor.

```
 1: <Directory /home/*/public_html>
 2:  AllowOverride FileInfo AuthConfig Limit
 3:  Options MultiViews Indexes SymLinksIfOwnerMatch IncludesNoExec
 4:  <Limit GET POST OPTIONS PROPFIND>
 5:   Order Allow,Deny
 6:   Allow from all
 7:  </Limit>
 8:  <LimitExcept GET POST OPTIONS PROPFIND>
 9:   Order Deny,Allow
10:   Deny from all
11:  </LimitExcept>
12: </Directory>
```

În secţiunea următoare, veţi învăţa despre restricţionarea accesului de partea aplicaţiei client, pe baza informaţiilor stocate în cookie-uri.

O prezentare a cookie-urilor

De partea aplicaţiei, puteţi folosi cookie-uri în cadrul scripturilor dumneavoastră PHP pentru a controla anumite zone ale serverului dumneavoastră Web. Un *cookie* (prăjiturică) este un mic pachet de date stocate de către browserul utilizatorului în urma unei solicitări din partea unui server sau script. Un site poate să ceară ca browserul să stocheze până la 20 de cookie-uri. Fiecare dintre cookie-uri constă dintr-un nume, o valoare, o dată de expirare, precum şi din informaţii despre site-ul şi calea de care aparţine. Un cookie nu poate fi mai mare de 4 KB.

După ce cookie-ul este creat, numai site-ul care l-a solicitat poate să citească datele stocate în el, fapt care asigură respectarea dreptului la intimitate al utilizatorului. Mai mult, utilizatorul poate să îşi configureze browserul în asa fel încât să îl anunţe despre cookie-urile care sunt create şi poate să refuze complet toate solicitările de creare a cookie-urilor. Din acest motiv, cookie-urile trebuie folosite cu moderaţie şi nu ar trebui să folosiţi cookie-uri pe post de element de bază al întregului proiect al aplicaţiei dumneavoastră fără a-l avertiza pe utilizator în acest sens.

Anatomia unui cookie

Un script PHP care creează un cookie ar putea trimite anteturi precum cele de mai jos:

```
HTTP/1.1 200 OK
Date: Tue, 26 Aug 2003 13:39:58 GMT
Server: Apache/2.0.47 (Unix) PHP/4.3.3
X-Powered-By: PHP/4.3.3
```

```
Set-Cookie: vegetable=artichoke; path=/; domain=yourdomain.com
Connection: close
Content-Type: text/html
```

După cum puteți vedea, acest antet `Set-Cookie` conține o pereche nume / valoare, o cale și un nume de domeniu. Numele și valoarea vor fi codificate la fel ca URL-urile. În cazul în care este prezent, câmpul `expires` este o instrucțiune care îi spune browserului să „uite" cookie-ul după o anumită dată și o anumită oră. Câmpul `path` definește poziția din cadrul unui site Web de la nivelul căreia browserul trebuie să trimită înapoi cookie-ul către server odată cu fiecare solicitare. Câmpul `domain` stabilește domeniile de Internet către care trebuie trimis cookie-ul. Domeniul nu poate să fie diferit de domeniul de la care a fost trimis cookie-ul, dar există, totuși, un grad oarecare de flexibilitate. În exemplul precedent, browserul va trimite cookie-ul către serverul `yourdomain.com`, dar și către serverul `www.yourdomain.com`.

Dacă browserul este configurat să stocheze cookie-uri, acesta va păstra informațiile până la data expirării lor. Dacă utilizatorul vizitează orice pagină care respectă condițiile specificate de calea și domeniul unui cookie, browserul va trimite cookie-ul către server. Anteturile trimise de browser ar putea să arate similar cu cele de mai jos:

```
GET / HTTP/1.Ø
Connection: Keep-Alive
User-Agent: Mozilla/4.Ø (compatible; MSIE 6.Ø; Windows 98)
Host: www.yourdomain.com
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, image/png, */*
Accept-Encoding: gzip
Accept-Language: en,pdf
Accept-Charset: iso-8859-1,*,utf-8
Cookie: vegetable=artichoke
```

Un script PHP va avea, apoi, acces la cookie prin intermediul variabilei de mediu de operare `HTTP_COOKIE` sau prin intermediul vectorului superglobal `$_COOKIE`:

```
echo "$_SERVER[HTTP_COOKIE]<BR>";  // afiseaza "vegetable=artichoke"
echo getenv("HTTP_COOKIE")."<BR>"; // afiseaza "vegetable=artichoke"
echo $_COOKIE['vegetable']."<BR>"; // afiseaza "artichoke"
```

Crearea unui cookie în PHP

Puteți crea un cookie în cadrul unui script PHP în două moduri. Pe de-o parte, puteți folosi funcția `header()` pentru a transmite către browser antetul `Set-Cookie`. Funcția `header()` primește drept argument un șir de caractere ce va fi inclus în secțiunea de anteturi trimise de server. Deoarece anteturile sunt trimise automat, funcția `header()` trebuie apelată înainte ca orice fel de date să fie trimise către browser:

```
header ("Set-Cookie: vegetable=artichoke; expires=Wed, 27-Aug-Ø3 14:39:58 GMT;
path=/; domain=yourdomain.com");
```

Deşi nu este dificilă, această metodă de trimitere a unui cookie implică folosirea unei funcţii care să construiască şirul de caractere ce va fi trimis drept antet. Formatarea datei, ca în acest exemplu, sub forma unei perechi nume / valoarea codificată URL, nu este o sarcină prea plăcută. Ar putea fi însă un exerciţiu de reinventare a roţii, deoarece PHP ne oferă o funcţie care face toate aceste lucruri în locul nostru.

Funcţia `setcookie()` face exact ceea ce îi spune numele — trimite către browser un antet `Set-Cookie`. Din această cauză, ea trebuie apelată înainte de trimiterea oricăror altor date către browser. Funcţia acceptă drept argumente numele cookie-ului, valoarea cookie-ului, data de expirare în format de *timestamp* Unix, calea, domeniul şi un număr întreg care trebuie să fie 1 în cazul în care cookie-ul trebuie să fie trimis numai prin intermediul unei conexiuni securizate. Toate argumentele acestei funcţii sunt opţionale cu excepţia primului (numele cookie-ului).

Listingul 23.6 foloseşte funcţia `setcookie()` pentru a crea un cookie.

LISTINGUL 23.6. Crearea unui cookie şi afişarea valorii acestuia.

```
 1: <?php
 2: setcookie("vegetable", "artichoke", time()+3600, "/", "yourdomain.com", 0);
 3: ?>
 4: <html>
 5: <head>
 6: <title>Listing 23.6 Setting and printing a cookie value</title>
 7: </head>
 8: <body>
 9: <?php
10: if (isset($_COOKIE[vegetable])) {
11:  echo "<p>Hello again, your chosen vegetable is $_COOKIE[vegetable]</p>";
12: } else {
13:  echo "<p>Hello you. This may be your first visit</p>";
14: }
15: ?>
16: </body>
17: </html>
```

Deşi creăm cookie-ul pe linia 2 atunci când scriptul este rulat prima oară, variabila `$_COOKIE['vegetable']` nu va fi creată şi ea în acelaşi moment. Cookie-ul este citit numai atunci când este trimis de către browser către server. Acest lucru nu se va întâmpla decât în momentul în care un utilizator vizitează din nou o pagină din cadrul domeniului dumneavoastră. Pe linia 2, creăm un cookie cu numele `"vegetable"` şi cu valoarea `"artichoke"` (anghinare). Folosim funcţia `time()` pentru a afla data şi ora curentă şi adăugăm 3600 valorii obţinute (o oră are 3600 de secunde). Suma rezultată reprezintă data de expirare. Definim calea `"/"`, ceea ce înseamnă că acest cookie trebuie trimis către server odată cu solicitarea oricărei pagini din cadrul site-ului nostru. Atribuim argumentului `domain` valoarea `"yourdomain.com"`, ceea ce înseamnă că acest cookie trebuie trimis oricărui server (subdomeniu) din acest domeniu. Ultimului argument pe care îl pasăm funcţiei `setcookie()` îi dăm valoarea 0, ceea ce înseamnă că acest cookie poate fi trimis şi prin conexiuni care nu sunt securizate.

Pasarea unui şir vid ("") în locul unui argument de tip şir de caractere, sau a valorii Ø pentru un argument ce trebuie să fie un număr întreg, face ca funcţia `setcookie()` să ignore argumentul respectiv.

Apropo

Atunci când folosim date de expirare create dinamic într-un cookie, la fel ca în exemplul precedent, acestea sunt obținute prin adăugarea unui număr de secunde timpului curent al sistemului care rulează Apache sau PHP. Dacă ceasul sistemului nu este corect, este posibil să îi atribuim cookie-ului o dată de expirare care a trecut deja.

Ştergerea unui cookie

Oficial, pentru a şterge un cookie, trebuie să apelăm funcţia `setcookie()` pasându-i doar argumentul corespunzător numelui cookie-ului:

```
setcookie("vegetable");
```

Această metodă nu funcţionează întotdeauna cum trebuie, însă, şi ar fi mai bine să nu vă bazaţi pe ea. Cea mai sigură metodă este să creaţi un cookie folosind o dată de care sunteţi sigur că a expirat.

```
setcookie("vegetable", "", time()-60, "/", "yourdomain.com", 0);
```

Ar mai trebui să vă asiguraţi şi că pasaţi funcţiei `setcookie()` aceeaşi cale, domeniu şi parametru de securitate ca cele trimise atunci când aţi creat iniţial cookie-ul.

Restricționarea accesului pe baza valorilor unui cookie

Haideţi să trecem, acum, la lucruri interesante — folosirea abilităţilor dumneavoastră de crearea a cookie-urilor pentru restricţionarea accesului la anumite părţi ale site-ului dumneavoastră Web! Să presupunem că aţi creat un formular de autentificare şi aţi verificat valorile introduse faţă de cele dintr-o bază de date. Dacă utilizatorul este autentificat, îi trimiteţi un cookie ce spune că aceste lucruri s-au întâmplat. Apoi, pentru toate paginile pe care vreţi să le faceţi disponibile doar utilizatorilor autorizaţi, verificaţi existenţa respectivului cookie. Dacă cookie-ul nu este prezent, utilizator este fie trimis înapoi la formularul de autentificare, fie este afişat pe ecran un mesaj despre restricţionarea accesului.

Vom parcurge fiecare dintre aceste etape în cadrul următoarelor câteva secţiuni.

Crearea tabelei cu utilizatori autorizați

Atunci când implementaţi conceptul de conturi de utilizator într-o aplicaţie Web, practica uzuală este ca informaţiile specifice ale utilizatorilor să fie stocate într-o tabela a unei baze de date.

Informaţiile din această tabelă pot fi folosite, apoi, pentru autorizarea utilizatorului şi acordarea accesului la secţiunile care sunt concepute pentru aceşti utilizatori „speciali".

Următoarea comandă de creare a unei tabele va crea o tabelă denumită `auth_users` în baza dumneavoastră de date MySQL, conţinând câmpuri pentru ID, prenume, nume de familie, adrese de e-mail şi parolă:

```
mysql> create table auth_users (
 -> id int not null primary key auto_increment,
 -> f_name varchar(50),
 -> l_name varchar(50),
 -> email varchar(150),
 -> username varchar(25),
 -> password varchar (75)
 ->);
Query OK, 0 rows affected (0.03 sec)
```

Următoarea comandă `INSERT` introduce o înregistrare în tabela `auth_users` pentru un utilizator numit `John Doe`, având adresa de e-mail `john@doe.com`, numele de utilizator `jdoe` şi parola `doepass`:

```
mysql> insert into auth_users values ('', 'John', 'Doe', 'john@doe.com',
 -> 'jdoe', password('doepass'));
Query OK, 1 row affected (0.00 sec)
```

Această comandă `INSERT` ar trebui să fie extrem de uşor de înţeles, cu excepţia folosirii funcţiei `password()`. Atunci când folosim această funcţie în comanda `INSERT`, valoarea stocată în tabelă nu este parola efectivă, ci un hash al parolei (o valoare codificată).

Atunci când vizualizaţi conţinutul tabelei `auth_users`, veţi vedea hash-ul în câmpul corespunzător parolei, după cum urmează:

```
mysql> select * from auth_users;
+----+--------+--------+--------------+----------+------------------+
| id | f_name | l_name | email        | username | password         |
+----+--------+--------+--------------+----------+------------------+
| 1  | John   | Doe    | john@doe.com | jdoe     | 2fae5c9d478ec4b1 |
+----+--------+--------+--------------+----------+------------------+
1 row in set (0.01 sec)
```

Deşi pare că este vorba despre criptare, un hash nu este, de fapt, o modalitate de criptare a informaţiilor. Este vorba despre o „amprentă" a informaţiei originale. Hash-urile sunt folosite, în general, pentru a stabili autenticitatea unei informaţii. În acest caz, atunci când verificaţi parola utilizatorului dumneavoastră, veţi verifica dacă parola introdusă se potriveşte cu hash-ul stocat. Folosirea hash-urilor reduce necesitatea — şi riscurile de securitate — de a stoca parolele efective.

Crearea formularului şi scriptului de autentificare

După ce aţi autorizat utilizatorii în tabela dumneavoastră, trebuie să le oferiţi un mecanism prin care să îşi demonstreze autenticitatea. În acest caz, nu vom avea nevoie decât de un formular simplu cu două câmpuri, aşa cum se vede în listingul 23.7.

LISTINGUL 23.7. Formularul pentru autentificarea utilizatorilor.

```
 1: <html>
 2: <head>
 3: <title>Listing 23.7 User Login Form</title>
 4: </head>
 5: <body>
 6: <H1>Login Form</H1>
 7: <FORM METHOD="POST" ACTION="listing23.8.php">
 8: <P><STRONG>Username:</STRONG><BR>
 9: <INPUT TYPE="text" NAME="username"></p>
10: <P><STRONG>Password:</STRONG><BR>
11: <INPUT TYPE="password" NAME="password"></p>
12: <P><INPUT TYPE="SUBMIT" NAME="submit" VALUE="Login"></P>
13: </FORM>
14: </body>
15: </html>
```

Puneţi aceste linii într-un fişier text denumit `listing23.7.php` şi amplasaţi-l în directorul rădăcină al serverului dumneavoastră Web. În listingul 23.8 veţi crea scriptul efectiv, pe care formularul îl apelează aşteptându-se la numele `listing23.8.php`.

LISTINGUL 23.8. Scriptul pentru autentificarea utilizatorilor.

```
 1: <?php
 2: //verifica existenta campurilor obligatorii ale formularului
 3: if ((!$_POST[username]) || (!$_POST[password])) {
 4:  header("Location: listing23.7.php");
 5:  exit;
 6: }
 7:
 8: //se conecteaza la server si selecteaza baza de date
 9: $conn = mysql_connect("localhost", "joeuser", "somepass")
10:  or die(mysql_error());
11: mysql_select_db("testDB",$conn) or die(mysql_error());
12:
13: //creează si lanseaza interogarea
14: $sql = "select f_name, l_name from auth_users where username =
15: '$_POST[username]' AND password = password('$_POST[password]')";
16: $result = mysql_query($sql,$conn) or die(mysql_error());
17:
18: //preia numarul de linii al setului de rezultate; ar trebui sa fie 1
19: if (mysql_num_rows($result) == 1) {
20:
21: //daca este autorizat, preia valorile lui f_name si l_name
22: $f_name = mysql_result($result, 0, 'f_name');
23: $l_name = mysql_result($result, 0, 'l_name');
```

LISTINGUL 23.8. (*continuare*)

```
24:
25: //configureaza cookie-ul de autorizare
26: setcookie("auth", "1", 0, "/", "yourdomain.com", 0);
27:
28: //pregateste mesajul pentru a fi afisat, precum si meniul
29: $msg = "<P>$f_name $l_name is authorized!</p>";
30: $msg .= "<P>Authorized Users' Menu:";
31: $msg .= "<ul><li><a href=\"listing23.9.php\">secret page</a></ul>";
32:
33: } else {
34:
35: //redirectioneaza la formularul de autentificare in caz contrar
36: header("Location: listing23.7.php");
37: exit;
38: }
39: ?>
40: <HTML>
41: <HEAD>
42: <TITLE>Listing 23.8 User Login</TITLE>
43: </HEAD>
44: <BODY>
45: <? print "$msg"; ?>
46: </BODY>
47: </HTML>
```

Puneţi aceste linii într-un fişier text denumit `listing23.8.php` şi amplasaţi acest fişier în directorul rădăcină al serverului dumneavoastră Web. În câteva momente îl veţi încerca, dar haideţi, mai întâi, să analizăm ce face acest script.

Linia 3 verifică existenţa celor două câmpuri obligatorii din formular. Acestea sunt şi singurele câmpuri ale formularului: `username` şi `password`. Dacă oricare dintre aceste câmpuri este absent, scriptul va redirecţiona utilizatorul înapoi la formularul de autentificare. Dacă cele două câmpuri sunt prezente, scriptul trece mai departe la liniile 9–11 care stabilesc conexiunea cu serverul de baze de date şi selectează baza de date ce trebuie folosită, pregătind astfel emiterea interogării SQL care să verifice autenticitatea utilizatorului. Această interogare şi executarea ei se găsesc pe liniile 14–16. Observaţi că interogarea verifică echivalenţa dintre hash-ul parolei introduse în formular şi cel stocat în baza de date. Aceste două elemente trebuie să fie identice şi, în acelaşi timp, să aparţină utilizatorului în chestiune pentru ca utilizatorul să fie autentificat.

Linia 19 verifică rezultatul interogării numărând rândurile care se găsesc în setul de rezultate. Numărul de rânduri trebuie să fie exact 1 dacă perechea nume de utilizator / parolă reprezintă o acreditare validă. În acest caz, funcţia `mysql_result()` este folosită pe liniile 22–23 pentru a extrage prenumele şi numele de familie al utilizatorului. Aceste nume nu au decât un scop estetic. Linia 26 creează un cookie de autorizare. Numele cookie-ului este `auth`, iar valoarea este 1. Dacă argumentul referitor la perioada de valabilitate este 0, cookie-ul va rămâne valabil atâta vreme cât sesiunea de lucru cu browserul este deschisă. Atunci când

utilizatorul închide browserul, cookie-ul va expira. Liniile 29–31 creează un mesaj de afişat, care include un link către un fişier pe care îl vom crea imediat.

La final, liniile 33–38 tratează o tentativă eşuată de autentificare. În acest caz, utilizatorul este pur şi simplu redirecţionat către formularul de autentificare.

Accesaţi formularul de autentificare şi introduceţi valorile corecte pentru utilizatorul `John Doe`. Atunci când transmiteţi formularul, rezultatul ar trebui să arate ca în figura 23.1.

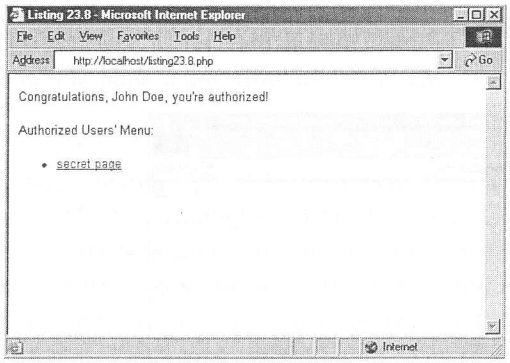


FIGURA 23.1. Rezultatul unei autentificări reuşite.

Încercaţi să vă autentificaţi folosind un o pereche invalidă de nume de utilizator şi parolă; ar trebui să fiţi redirecţionat înapoi la formularul de autentificare. În secţiunea următoare, care este şi ultima, veţi crea scriptul `listing23.9.php` care va citi cookie-ul de autentificare pe care tocmai l-aţi creat şi se va comporta ca atare.

Verificarea cookie-ului `auth`

Ultima piesă a acestui puzzle este reprezentată de folosirea cookie-ului `auth` pentru a îi permite unui utilizator accesul la un fişier privat. În acest caz, fişierul privat este afişat în listingul 23.9.

LISTINGUL 23.9. Verificarea cookie-ului `auth`.

```
1: <?php
2: if ($_COOKIE[auth] == "1") {
3:  $msg = "<p>You are an authorized user.</p>";
4: } else {
5: //redirectioneaza la formularul de autentificare daca nu este autentificat
6: header("Location: listing23.7.php");
7: exit;
8: }
9. ?>
10: <html>
```

LISTINGUL 23.9. *(continuare)*

```
11: <head>
12: <title>Listing 23.9 Accessing a restricted page </title>
13: </head>
14: <body>
15: <?php print "$msg"; ?>
16: </body>
17: </html>
```

Din meniul afişat în figura 23.1, daţi clic pe linkul `secret`. Deoarece aţi devenit un utilizator autorizat, ar trebui să vedeţi un rezultat asemănător cu cel din figura 23.2.

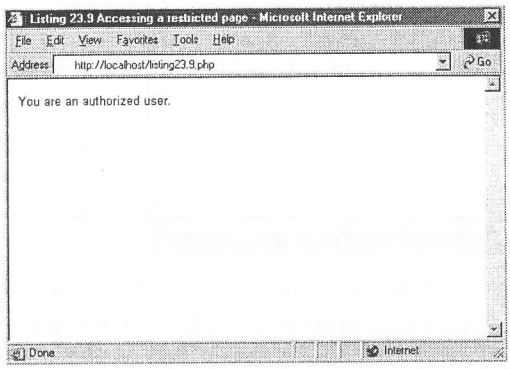


FIGURA 23.2. Accesarea unei pagini secrete în calitate de utilizator autorizat.

Închideţi browserul şi încercaţi să accesaţi fişierul `listing23.9.php` direct. Veţi descoperi că nu puteţi să faceţi acest lucru şi veţi fi redirecţionat către formularul de autentificare, deoarece cookie-ul nu există.

Rezumat

Acest capitol v-a explicat cum să folosiţi Apache pentru restricţiona accesul la site-ul dumneavoastră Web pe baza identităţii utilizatorului, pe baza informaţiilor din solicitarea HTTP sau pe baza caracteristicilor conexiunii. De asemenea, au mai fost discutate câteva module de autentificare incluse în pachetul Apache, precum şi câteva instrumente pe care le puteţi folosi pentru a crea şi gestiona baze de date cu utilizatori şi grupuri de utilizatori.

Pe lângă acestea, vi s-a prezentat modul de folosire a cookie-urilor şi aţi învăţat să utilizaţi funcţia `setcookie()` pentru a crea cookie-uri în browserul utilizatorului dumneavoastră. Aţi învăţat să folosiţi valorile stocate în cookie-uri pentru a permite accesul la anumite zone din aplicaţia dumneavoastră PHP.

Întrebări şi răspunsuri

Î. Am un sistem care rulează Unix. Pot să folosesc fişierul /etc/passwd ca bază de date cu utilizatori?

R. Deşi folosirea fişierului `/etc/passwd` poate părea mai convenabilă, ar fi mai bine să nu folosiţi fişierul existent `etc/passwd` pentru autentificarea utilizatorilor site-ului dumneavoastră Web. Altminteri, un atacator care dobândeşte accesul la datele de autentificare pentru serverul Web capătă acces şi la sistemul dumneavoastră. Păstraţi bazele de date separat şi încurajaţi-vă utilizatorii să aleagă parole diferite pentru contul de sistem şi cel de Web. Din când în când, folosiţi verificatoare de parole care scanează după parole slabe şi utilizatori ale căror parole sunt identice cu parolele.

Î. De ce mi se cere parola de două ori atunci când vizitez unele site-uri Web?

R. Browserul ţine minte parolele astfel încât să nu trebuiască să le tastaţi de fiecare dată când trimiteţi solicitări HTTP. Parolele stocate se bazează pe directiva `AuthName` şi pe numele host-ului serverului Web. Uneori puteţi să accesaţi un acelaşi site Web prin intermediul mai multor nume, cum ar fi `yourdomain.com` şi `www.yourdomain.com`. Dacă sunteţi autorizat să accesaţi o anumită zonă restricţionată de pe site-ul `yourdomain.com`, dar sunteţi redirecţionat sau urmaţi un link către `www.yourdomain.com`, vi se va cere din nou să furnizaţi un nume de utilizator şi o parolă deoarece browserul crede că este vorba despre un site complet diferit.

Î. Există vreo problemă serioasă de securitate sau de încălcare a intimităţii creată de folosirea cookie-urilor?

R. Un server poate să acceseze doar cookie-urile care au fost create de către propriul său domeniu. Deşi cookie-urile pot fi stocate pe discul dur al utilizatorului, nu există nici o altă interacţiune cu sistemul de fişiere al utilizatorului. Este posibil, totuşi, ca un cookie să fie creat ca răspuns la solicitarea unei imagini de pe server. De aceea, dacă multe site-uri includ imagini servite de o terţă parte sau de un server de reclame sau de un script de contorizare a numărului de vizitatori, această terţă parte ar putea să urmărească un utilizator dintr-un domeniu într-altul.

Atelier

Atelierul este conceput să anticipeze posibile întrebări, să vă ajute să recapitulaţi cele învăţate şi să începeţi să vă puneţi cunoştinţele în practică.

Test

1. Care sunt avantajele fişierelor de tip bază de date faţă de fişierele text în ceea ce priveşte stocarea informaţiilor de autentificare a utilizatorilor?

2. Puteţi să enumeraţi câteva dezavantaje ale autentificării HTTP de bază?
3. Ce funcţie este concepută pentru a vă permite să creaţi un cookie în browserul unui vizitator?

Soluţii

1. Fişierele de tip bază de date sunt mai scalabile deoarece pot fi indexate. Acest lucru înseamnă că Apache nu trebuie să citească fişierul în mod secvenţial până ce găseşte înregistrarea corespunzătoare unui anumit utilizator, ci poate să sară direct la poziţia exactă a acesteia.
2. Unul dintre dezavantaje este acela că informaţia este trimisă în clar prin reţea. Acest lucru înseamnă că, în afara cazului în care folosiţi SSL, este posibil ca un atacator să citească pachetele pe care browserul dumneavoastră le trimite către server şi să vă fure parolele. Un alt dezavantaj este acela că autentificarea HTTP nu permite personalizarea interfeţei de autentificare (cu excepţia numelui zonei securizate). Este o practică uzuală ca site-urile Web să implementeze mecanisme proprii de autentificare folosind formulare HTML şi cookie-uri.
3. Funcţia `setcookie()` vă permite să creaţi un cookie (deşi aţi putea face acelaşi lucru trimiţând un header `Set-Cookie` cu ajutorul funcţiei `header()`).

Activitate

Exersaţi folosirea diverselor tipuri de autentificare pe site-ul dumneavoastră de teste — atât bazate pe server cât şi implementate în PHP. Obişnuiţi-vă cu diferenţele dintre autentificarea HTTP de bază şi metodele de autentificare pe care le creaţi dumneavoastră.

CAPITOLUL 24
Înregistrarea şi monitorizarea activităţii serverului Web

Acest capitol descrie modul de funcţionare al sistemului de înregistrare de fişiere jurnal pus la dispoziţie de Apache şi metodele prin care aceste fişiere jurnal pot fi personalizate, specificând ce informaţii să fie înregistrate şi unde să fie salvate. În plus, veţi mai învăţa şi să folosiţi PHP şi MySQL pentru a înregistra în jurnale anumite elemente care v-ar putea interesa şi care nu fac obiectul fişierelor jurnal generate de Apache.

În acest capitol, veţi:

- înţelege formatul fişierelor jurnal ale serverului Apache şi nivelurile de înregistrare;
- alterna şi analiza fişierele jurnal ale serverului Apache;
- interpreta erorile frecvente pe care le puteţi întâlni în fişierele jurnal;
- crea scripturi care să înregistreze anumite elemente în tabele ale bazelor de date;
- crea rapoarte personalizate pe baza tabelelor jurnal.

Jurnalul de acces standard al serverului Apache

Folosind funcţiile elementare de înregistrare a fişierelor jurnal pe care le are Apache, puteţi să urmăriţi cine vă vizitează site-ul înregistrând accesele la serverele care le găzduiesc. Puteţi să înregistraţi toate aspectele solicitărilor şi răspunsurilor, inclusiv adresele de IP ale clienţilor, numele de utilizator şi resursele accesate. Trebuie să parcurgeţi trei etape pentru a crea un jurnal al solicitărilor primite de server.

1. Definiţi ce doriţi să înregistraţi — formatul jurnalului dumneavoastră.
2. Definiţi unde doriţi să înregistraţi — fişierul jurnal, baza de date sau aplicaţia externă
3. Definiţi dacă să înregistraţi sau nu — cu ajutorul regulilor condiţionale de înregistrare.

Stabilirea datelor ce trebuie înregistrate

Puteţi să înregistraţi în jurnale aproape toate aspectele asociate solicitărilor pe care le primeşte serverul. Puteţi să definiţi cum să arate intrările în jurnal creând un format de jurnal. Un format de jurnal este un şir de caractere ce conţine text şi directive de formatare a jurnalului. Directivele de formatare a jurnalului încep cu caracterul `%` şi sunt urmate de numele sau identificatorul directivei, care este de obicei o literă indicând tipul de informaţie ce trebuie înregistrată. Atunci când Apache înregistrează în jurnal o solicitare, şirul de caractere este scanat, iar fiecare dintre directive este înlocuită cu valoarea respectivă. De exemplu, dacă formatul jurnalului este `Aceasta este adresa clientului %a`, o intrare în acest jurnal va fi ceva de genul: `Aceasta este adresa clientului 10.0.0.2`. Cu alte cuvinte, directiva `%a` este înlocuită cu adresa de IP a clientului care a efectuat solicitarea. Tabelul 24.1 conţine o listă completă a directivelor de formatare disponibile.

TABELUL 24.1. Directivele de formatare a jurnalului.

Opţiuni de fomatare	Explicaţie
Date despre client	
`%a`	Adresa de IP obţinută de la programul client
`%h`	Numele host-ului sau adresa de IP a clientului care a trimis solicitarea. Numele host-ului va putea fi înregistrat doar dacă sunt îndeplinite două condiţii: adresa de IP a clientului trebuie să poată fi rezolvată la un nume de domeniu prin intermediul unei interogări DNS inversate, iar Apache trebuie să fie configurat pentru a efectua această intergoare cu ajutorul directivei `HostNameLookups`, explicată mai jos în acest capitol. Dacă aceste condiţii nu sunt îndeplinite, adresa de IP a clientului nu va fi înregistrată în jurnal.
`%l`	Numele utilizatorului autentificat pe calculatorul client, obţinut prin protocolul `identd`. Această opţiune nu este foarte utilă, deoarece acest protocol nu este implementat pe majoritatea calculatoarelor şi pentru că rezultatele nu sunt, oricum, de încredere pentru că sunt furnizate de un program care rulează pe calculatorul utilizatorului.
`%u`	Numele utilizatorului autentificat prin protocolul HTTP de bază.
Date despre server	
`%A`	Adresa de IP locală, de la server.
`%D`	Timpul necesar pentru rezolvarea solicitării, în microsecunde.
`%{var_de_mediu}e`	Valoarea unei variabile a mediului de operare al cărei nume este înlocuit în câmpul *var_de_mediu*.
`%{format_calendaristic}t`	Data şi ora curentă. Dacă *{format_calendaristic}* este prezent, acesta va fi interpretat ca argument pasat funcţiei Unix `strftime`. Consultaţi şi manualul Apache, pagina `logresolve`, pentru mai multe detalii.
`%T`	Timpul necesar îndeplinirii acţiunii solicitate, în secunde.
`%v`	Numele corect al serverului care a răspuns solicitării respective.

TABELUL 24.1. (*continuare*)

Opţiuni de fomatare	Explicaţie
Date despre server	
%V	Numele serverului, pe baza directivei `UserCanonicalName`.
%X	Starea conexiunii din punctul de vedere al serverului. Valoarea x înseamnă că a fost abandonată conexiunea înainte ca serverul să poată trimite datele. Valoarea + înseamnă că a fost păstrată activă pentru solicitări ulterioare de la acelaşi client. Valoarea – înseamnă că va fi închisă conexiunea.
Date despre solicitare	
%{*nume_cookie*}C	Valoarea unui cookie cu numele *nume_cookie*.
%H	Protocolul în care a fost făcută solicitarea, de exemplu HTTP sau HTTPS.
%m	Metoda folosită de solicitare, cum ar fi `GET`, `POST`, `PUT` şi aşa mai departe.
%{*nume_antet*}i	Valoarea antetului cu numele *nume_antet* din solicitarea primită de la client. Aceste informaţii pot fi utile, de exemplu, pentru a înregistra numele şi versiunile browserelor folosite de vizitatorii dumneavoastră.
%r	Textul solicitării HTTP originale.
%q	Parametrii interogării HTTP, dacă există.
%U	URL-ul solicitat, fără parametrii interogării.
%y	Numele de utilizator folosit la autentificarea HTTP (de bază sau de tip digest).
Date despre răspuns	
%b, %B	Dimensiunea, în octeţi, a corpului răspunsului trimis înapoi către client (mai puţin anteturile). Singura diferenţă între cele două opţiuni este că, în cazul în care nu au fost trimise nici un fel de date, %b va înregistra caracterul –, iar %B va înregistra caracterul Ø.
%f	Calea către fişierul servit, în cazul în care este servit vreun fişier.
%t	Ora la care a fost îndeplinită solicitarea.
%{*nume_antet*}o	Valoarea unui antet cu numele *nume_antet* din răspunsul serverului către client.
%>s	Codul final de stare. Apache poate procesa de mai multe ori o aceeaşi solicitare (în cazul în care au loc redirecţionări interne). Acesta este codul de stare al răspunsului final.

CLF (Common Log Format) reprezintă un standard de format pentru fişierele jurnal. Cele mai multe site-uri Web pot înregistra solicitările în fişiere jurnal folosind acest format, iar acest format este înţeles de majoritatea utilitarelor de procesare a fişierelor jurnal şi de generare de raporturi. Acest format este următorul:

```
"%h %l %u %t \"%r\" %>s %b"
```

Cu alte cuvinte, sunt incluse numele host-ului sau adresa de IP a clientului, utilizatorul autentificat pe calculatorul client, obţinut prin `identd`, numele de utilizator din autentificarea HTTP, data şi ora la care a fost îndeplinită solicitarea, textul solicitării, codul de stare şi dimensiunea în octeţi a datelor servite.

Apropo

> Puteţi să citiţi documentaţia Common Log Format de pe site-ul original al organizaţiei W3C, la adresa `http://www.w3c.org/Daemon/User/Config/Logging.html`.

Următoarea linie ilustrează o intrare într-un fişier jurnal în format CLF:

```
10.0.0.1 - - [26/Aug/2003:11z:27:56 -0800] "GET / HTTP/1.1" 200 1456
```

Acum sunteţi gata să definiţi formate pentru fişierele jurnal folosind directiva `LogFormat`. Această directivă acceptă două argumente: primul argument este un şir de formatare a jurnalului, iar cel de-al doilea este o poreclă care va fi asociată cu acel şir de formatare.

De exemplu, următoarea directivă din fişierul original de configurare al serverului Apache defineşte formatul CLF şi îi asociază porecla `common`:

```
LogFormat "%h %l %u %t \"%r\" %>s %b" common
```

Puteţi să mai folosiţi directiva `LogFormat` şi cu un singur argument, fie un şir de formatare, fie o poreclă. Acest lucru va avea drept efect stabilirea valorii implicite a formatului folosit de directiva `TransferLog`, care este explicată în secţiunea *Salvarea jurnalelor în fişiere*, prezentată începând cu pagina 454 în acest capitol.

Directiva `HostNameLookups`

Atunci când un client transmite o solicitare, Apache ştie doar adresa de IP a clientului. Apache trebuie să efectueze o operaţiune denumită interogare DNS inversată, pentru a afla numele de host asociat cu respectiva adresă de IP. Această operaţiune poate dura mult şi poate cauza întârziere semnificativă a răspunsului la solicitare. Directiva `HostNameLookups` vă permite să controlaţi dacă aceste interogări DNS inversate sunt sau nu efectuate.

`HostNameLookups` preia următoarele valori drept argument: `on`, `off`, `double`. Valoarea implicită este `off`. Interogarea de tip `double` înseamnă că Apache va încerca să găsească numele host-ului pe baza adresei de IP şi, apoi, va încerca să afle adresa de IP pe baza numelui de host descoperit. Acest proces este necesar dacă securitatea vă interesează foarte mult, conform descrierii de la adresa `http://httpd.apache.org/docs-2.0/dns-caveats.html`. Dacă folosiţi numele de domeniu în cadrul directivelor `Allow` şi `Deny`, interogarea DNS dublă este efectuată indiferent de valoarea directivei `HostNameLookups`.

Dacă `HostNameLookups` este `on` sau `double`, Apache va înregistra numele host-urilor în jurnale. Acest lucru încarcă suplimentar serverul, lucru pe care trebuie să îl luaţi în seamă atunci când hotărâţi dacă să activaţi sau nu directiva `HostNameLookups`.

Dacă alegeţi să dezactivaţi directiva `HostNameLookups`, variantă recomandată pentru site-urile cu trafic mediu şi mare, Apache va înregistra în jurnal numai adresele de IP asociate. Există multe utilitare cu care puteţi, apoi, să rezolvaţi numele host-urilor pe baza adreselor de IP. Consultaţi secţiunea *Gestionarea jurnalelor generate de Apache*, prezentată începând cu pagina 457 în acest capitol. În plus, rezultatul va fi pasat scripturilor de tip CGI prin intermediul variabilei de mediu de operare `REMOTE_HOST`.

Directiva `IdentityCheck`

La începutul acestui capitol, am explicat cum se înregistrează numele de utilizator obţinut prin protocolul identd, folosind directiva de formatare `%l`. Directiva `IdentityCheck` preia drept argument valoarea `on` sau `off` pentru a activa sau dezactiva verificarea acestei identităţi, astfel încât să fie disponibilă pentru includerea în jurnale. Din cauză că această informaţie nu este de încredere şi pentru că verificarea ei durează mult, directiva are implicit valoarea `off` şi, probabil, nici nu ar trebui activată vreodată. Am menţionat `%l` doar pentru că face parte din formatul CLF. Pentru mai multe informaţii despre protocolul `identd`, consultaţi RFC 1413 la adresa `http://www.rfc-editor/rfc/rfc1413.txt`.

Variabilele de mediu de operare

Directiva `CustomLog` acceptă o variabilă de mediu de operare ca un al treilea argument. Dacă variabila de mediu de operare este prezentă, intrarea va fi înregistrată în jurnal; altminteri, intrarea nu va fi înregistrată. Dacă variabila este negată prin folosirea prefixului `!`, intrarea va fi înregistrată în jurnal doar dacă variabila nu este prezentă.

Următorul exemplu arată cum să evitaţi înregistrarea în jurnalele dumneavoastră a imaginilor în format GIF şi JPEG:

```
SetEnvIf Request_URI "(\.gif|\.jpg)$" image
CustomLog logs/access_log common env=!image
```

> Expresia regulată folosită pentru găsirea unui model, aici sau în orice altă parte a fişierului `httpd.conf`, respectă acelaşi format ca expresiile regulate din PHP sau din alte limbaje de programare.

Apropo

Codul de stare

Puteţi specifica dacă să fie înregistrate în intrările din jurnale anumite elemente. La începutul acestui capitol aţi învăţat că directivele de formatare încep cu caracterul `%`, urmat de un identificator al directivei. Între acestea două puteţi introduce o listă de coduri de stare, separate prin virgule. Dacă starea solicitării este una dintre cele enumerate, parametrul va fi înregistrat în jurnal; altminteri, va fi înregistrat un caracter –.

De exemplu, identificatorul de directivă `%400,501{User-agent}i` înregistrează numele şi versiunea browserului în cazul solicitărilor malformate (cod de stare 400) şi în cazul înregistrărilor conţinând metode ce nu sunt implementate (cod de stare 501). Aceste informaţii pot fi utile pentru a afla ce programe client provoacă probleme.

Puteţi folosi prefixul ! în faţa parametrului de format al jurnalului dacă metodele sunt implementate:

```
%!400,501{User-agent}i
```

Salvarea jurnalelor în fişiere

Salvarea jurnalelor în fişiere este modalitatea implicită de înregistrare a solicitărilor în Apache. Puteţi defini numele fişierului folosind directivele `TransferLog` şi `CustomLog`.

Directiva `TransferLog` preia numele fişierului drept argument şi foloseşte ultimul format definit de o directviă `LogFormat` cu un singur argument (porecla sau şirul de formatare). Dacă nu este prezent nici un format pentru jurnale, este folosit implicit formatul CLF.

Următorul exemplu arată cum se foloseşte directiva `LogFormat` şi directiva `TransferLog` pentru a defini un format de jurnal care este bazat pe CLF, dar mai include şi numele browserului:

```
LogFormat "%h %l %u %t \"%r\" %>s %b \"%{User-agent}i\""
TransferLog logs/access_log
```

Directiva `CustomLog` vă permite să specificaţi formatul de înregistrare a jurnalului în mod explicit. Ea preia cel puţin două argumente: un format de jurnal şi un fişier ţintă. Formatul jurnalului poate fi specificat sub forma unei porecle sau direct sub forma unui şir de formatare.

De exemplu, directivele:

```
LogFormat "%h %l %u %t \"%r\" %>s %b \"%{User-agent}i\"" myformat
CustomLog logs/access_log myformat
```

şi

```
CustomLog logs/access_log "%h %l %u %t \"%r\" %>s %b \"%{User-agent}i\""
```

sunt echivalente.

Directiva `CustomLog` poate prelua şi un al treilea argument opţional, sub forma unei variabile de mediu de operare, aşa cum s-a explicat în secţiunea *Variabilele de mediu de operare*, prezentată la pagina 453 în acest capitol.

Transmiterea înregistrărilor către o aplicaţie externă

Atât directiva `TransferLog` cât şi directiva `CustomLog` pot accepta drept argument o aplicaţie, prin folosirea prefixului `|`. Apache va scrie toate intrările în jurnal către portul interfaţa standard de intrare a programului. Programul va procesa, la rândul lui, intrările şi le va stoca într-o bază de date sau le va transmite către un alt sistem şi aşa mai departe.

Dacă programul se blochează, dintr-un motiv sau altul, Apache are grijă să îl repornească. Dacă serverul se opreşte, programul este oprit şi el.

Utilitarul `rotatelogs`, care se găseşte în pachetul Apache şi este explicat mai jos în acest capitol, este un exemplu de aplicaţie externă care preia jurnalele.

Regula generală este că, în afara cazului în care aveţi o nevoie specifică de a utiliza un anumit program, este mai uşor şi mai sigur să salvaţi jurnalele într-un fişier de pe disc şi să efectuaţi procesarea, centralizarea şi analiza jurnalelor ulterior, eventual pe un alt calculator.

Apropo

Asiguraţi-vă că programele pe care le folosiţi pentru înregistrarea solicitărilor sunt securizate, pentru că ele vor rula sub acreditarea utilizatorului cu care este pornit serverul Apache. Sub Unix, acesta este, de obicei, `root`, deoarece programul extern va fi pornit înainte ca Apache să îşi schimbe ID-ul de utilizator cu valoarea specificată de directiva `User`, care, de obicei, este `nobody`.

Înregistrarea standard a erorilor în jurnalele Apache

Apache poate fi configurat să înregistreze mesajele de eroare şi informaţiile de depanare într-un jurnal. În afară de erorile generate de Apache, vor fi înregistrate şi erorile CGI.

Fiecare intrare din jurnalul de erori este precedată de data şi ora la care s-a petrecut respectiva eroare şi de adresa de IP sau numele host-ului clientului, dacă acesta este disponibil. La fel ca şi în cazul solicitărilor HTTP, puteţi salva jurnalul într-un fişier sau îl puteţi trimite către o aplicaţie externă. Pe sistemele Unix puteţi, de asemenea, să folosiţi daemonul `syslog`. Modulele de Apache 1.3 care vă permit să înregistraţi erorile în jurnalul de evenimente al sistemului Windows vor fi, cu timpul, portate şi pentru Apache 2.0.

Puteţi folosi directiva ErrorLog pentru a defini unde vreţi să fie salvate jurnalele dumneavoastră. Această directivă preia un singur argument, care poate fi un fişier, un program sau daemonul `syslog`.

Salvarea jurnalelor de erori în fişiere

Pentru a specifica un fişier, trebuie să pasaţi în cadrul argumentului respectiv calea către fişier. Calea este interpretată relativ, adică se presupune că este relativă la directorul rădăcină al serverului. Fişierul jurnal de erori va fi, implicit, amplasat în directorul `logs` şi se va numi `error_log` sub Unix şi `error.log` sub Windows. Mai jos se găseşte un exemplu:

```
ErrorLog logs/my_error_log
```

Trimiterea erorilor către o aplicaţie externă

Puteţi specifica o cale către un program, precedată de prefixul `|`. Apache va trimite toate erorile către interfaţa standard de intrare a programului respectiv, iar programul va procesa, apoi, intrările respective. Mai jos, se găseşte un exemplu:

```
ErrorLog "|/usr/local/bin/someprogram"
```

Folosirea daemonului `syslog`

Pe un sistem Unix, dacă specificaţi `syslog` drept argument, puteţi înregistra jurnalul cu mesaje de eroare cu ajutorul daemonului Unix care se ocupă cu jurnalele sistemului, şi anume `syslogd`. Erorile înregistrate sunt salvate, implicit, în „incinta" (facility) corespunzătoare daemonului `syslog`, care este `local7`. Această incintă reprezintă o parte a sistemului care a generat eroarea. Puteţi specifica o incintă scriind argumentul sub forma `syslog:facility`. Exemple de incinte ce pot fi folosite de `syslog` sunt `mail`, `uucp`, `local0`, `local1` şi aşa mai departe. Pentru o listă completă, consultaţi documentaţia daemonului `syslog` care este inclusă în sistemul dumneavoastră (încercaţi comanda `man syslogd` sau `man syslog.conf` din linia de comandă). Mai jos se găseşte un exemplu de folosire a daemonului `syslog`:

```
ErrorLog syslog:local6
```

Directiva `LogLevel`

Informaţiile cu privire la erori care sunt furnizate de Apache sunt structurate în câteva categorii de importanţă. Puteţi alege să înregistraţi numai mesajele importante şi să treceţi cu vederea mesajele informative sau avertismentele triviale. Directiva `LogLevel` preia drept argument un nivel de importanţă al erorilor. Numai erorile de nivelul respectiv sau cu nivel mai mare de importanţă vor fi înregistrate în jurnal.

Tabelul 24.2 enumeră valorile valide pentru directiva `LogLevel`, aşa cum sunt ele specificate în documentaţia Apache. Valoarea implicită pentru `LogLevel` este `warn` (avertisment). Aceasta ar trebui să fie suficientă pentru majoritatea instalărilor de Apache. Dacă încercaţi să rezolvaţi o anumită problemă, aţi putea să folosiţi nivelul `debug` (depanare).

TABELUL 24.2. Opţiunile directivei `LogLevel` aşa cum sunt ele descrise în documentaţia serverului Apache.

Opţiune	Descriere	Exemplu
`emerg`	Urgenţe — sistemul nu poate fi folosit.	`Child cannot open lock file. Existing...`
`alert`	Trebuie luate imediat măsuri.	`getpwuid: couldn't determine user name from uid`
`crit`	Situaţie critică.	`socket: Failed to get a socket, exiting child`
`error`	Eroare.	`Premature end of script headers`
`warn`	Avertisment.	`Child process 1234 did not exit, sending another SIGHUP`
`notice`	Informaţie despre o situaţie normală dar semnificativă.	`httpd: caught SIGBUS, attempting to dump core in...`
`info`	Informaţie.	`Server seems busy, (You may need to increase StartServers or Min/MaxSpareServers)...`
`debug`	Mesaje utile pentru depanare.	`Opening config file...`

Gestionarea jurnalelor generate de Apache

Apache oferă câteva instrumente de gestiune a jurnalelor dumneavoastră. Există şi alte instrumente create de terţi în mod specific pentru Apache pe care le vom menţiona aici. Din moment ce Apache poate să înregistreze jurnalele în format CLF, majoritatea utilitarelor generice de procesare a jurnalelor pot fi folosite şi ele împreună cu Apache.

Rezolvarea numelor de host

Mai devreme în acest capitol aţi învăţat cum să folosiţi directiva `HostNameLookups` pentru a activa sau dezactiva rezolvarea numelor de host la momentul primirii solicitărilor. Dacă `HostNameLookups` este `off` (valoarea implicită), fişierul jurnal va conţine doar adresele de IP. Ulterior, puteţi folosi utilitarul din linie de comandă care se numeşte `logresolve` sub Unix şi `logresolve.exe` sub Windows pentru a procesa fişierul jurnal şi a converti adresele de IP în nume de host.

Utilitarul `logresolve` citeşte intrările de jurnal prin interfaţa de intrare standard şi trimite rezultatele sale către interfaţa de ieşire standard. Pentru a citi şi a scrie într-un fişier, trebuie să folosiţi redirecţionări în linia de comandă, atât în Unix cât şi în Windows:

```
logresolve < access.log > resolved.log
```

Instrumentele de rezolvare a adreselor din fişierele jurnal sunt eficiente deoarece ele pot să stocheze temporar rezultatele şi nu provoacă întârzieri ale răspunsului la solicitările clienţilor.

Alternarea fişierelor jurnal

În cazul site-urilor Web cu trafic foarte mare, fişierele jurnal pot creşte în dimensiuni foarte repede. Trebuie să aveţi un mecanism prin care să alternaţi fişierele jurnal periodic, arhivând şi comprimând fişierele mai vechi la intervale bine stabilite de timp.

Fişierele jurnal nu pot fi şterse direct atunci când Apache rulează deoarece serverul scrie direct în ele. Soluţia este folosirea unui program intermediar pentru a stoca intrările în jurnale. Programul va avea grijă, la rândul lui, de alternarea jurnalelor.

În acest scop, Apache ne pune la dispoziţie un utilitar care, sub Unix, se numeşte `rotatelogs`, iar sub Windows, `rotatelogs.exe`. Acest program acceptă trei argumente: un nume de fişier, un interval de alternare exprimat în secunde şi, opţional, un decalaj faţă de UTC (timpul universal coordonat) exprimat în minute.

De exemplu:

```
TransferLog "|bin/rotatelogs /var/logs/apachelog 86400"
```

va crea un nou fişier jurnal şi va muta fişierul curent în directorul `/var/logs` în fiecare zi. (Ultimul parametru al comenzii, 86400, este numărul de secunde dintr-o zi).

Apropo

> În cazul în care calea către program include spaţii, ar putea să trebuiască să evitaţi procesarea lor folosind caracterul de eludare \ (backslash) — de exemplu, `My\ Documents`. Această situaţie este foarte des întâlnită în cazul platformei Windows.

Dacă numele fişierului include opţiuni care folosesc prefixul `%`, numele va fi tratat şi când ar reprezenta un argument al funcţiei Unix `strftime` care converteşte opţiunile `%` în valori calendaristice şi orare. Pagina de manual a programului `rotatelogs` conţine o enumerare completă a acestor opţiuni, dar iată şi un exemplu:

```
TransferLog "|bin/rotatelogs /var/logs/apachelog%m_%d_%y 86400"
```

Această comandă va adăuga luna, ziua şi anul curent numelui fişierului jurnal.

Dacă numele nu include nici o opţiune de formatare `%`, timpul curent exprimat în secunde va fi adăugat numelui fişierului arhivat.

Unirea şi fragmentarea jurnalelor

Atunci când aveţi un grup de servere Web care servesc un conţinut similar, eventual situate „în spatele" unui server de echilibrare (load balancer), aveţi de foarte multe ori nevoie să uniţi jurnalele tuturor serverelor dumneavoastră într-unul singur înainte să îl pasaţi utilitarelor de analiză.

În mod similar, atunci când folosiţi o singură instanţă a serverului Apache pentru a deservi mai multe host-uri virtuale, uneori este util să fragmentaţi jurnalul în fişiere diferite, unul pentru fiecare dintre host-urile virtuale.

`Logtools` este o colecţie de instrumente de manipulare a fişierelor jurnal şi poate fi găsit la adresa `http://www.coker.com.au/logtools/`.

Apache include un script scris în Perl şi denumit `split-file` care realizează funcţia de fragmentare a fişierelor jurnal. Puteţi să găsiţi acest script în subdirectorul `support` al distribuţiei dumneavoastră de Apache.

Analizarea jurnalelor

După ce colectaţi jurnalele, puteţi să le analizaţi şi să obţineţi informaţii despre traficul dumneavoastră şi despre comportamentul vizitatorilor.

Sunt disponibile multe aplicaţii, atât comerciale cât şi gratuite, care vă pot ajuta să analizaţi jurnalele şi să creaţi rapoarte. Două dintre cele mai populare aplicaţii open source sunt Webalizer (`http://www.mrunix.net/webalizer/`) şi awstats (`http://awstats.sourceforge.net/`).

Wusage este o alternativă comercială, dar ieftină şi foarte simpatică, care poate fi găsită la adresa `http://www.boutell.com/wusage`.

Monitorizarea jurnalelor de erori

Dacă rulaţi Apache pe un sistem Unix, puteţi folosi şi utilitarul din linie de comandă `tail` pentru a monitoriza, în tip real, atât intrările din jurnalul de erori, cât şi pe cele din jurnalul de acces. Sintaxa este:

```
tail -f numefisier
```

unde `numefisier` este calea către fişierul jurnal generat de Apache. Acest utilitar va afişa pe ecran ultimele câteva linii din fişierul jurnal, iar apoi va continua să afişeze pe ecran fiecare linie pe măsură ce acestea sunt adăugate în fişier.

Puteţi găsi şi alte programe care vă permit să identificaţi rapid problemele scanând jurnalele dumneavoastră de erori pentru a găsi anumite erori, solicitări malformate şi aşa mai departe şi să generaţi, apoi, rapoarte:

- Logscan poate fi găsit la adresa `http://www.garandet.net/security.php`;
- ScanErrLog se află la adresa `http://www.librelogiciel.com/software`.

Înregistrarea jurnalelor personalizate într-o bază de date

Crearea propriilor dumneavoastră tabele jurnal în MySQL, împreună cu un pic de cod PHP, vă poate ajuta să capturaţi informaţii legate de accesul la anumite pagini ale site-ului dumneavoastră. Pe baza acestor informaţii, puteţi crea rapoarte personalizate. Această metodă se poate dovedi mult mai simplă decât descâlcirea fişierelor jurnal generate de Apache, în special atunci când căutaţi doar un anume subset de informaţii.

Crearea tabelei bazei de date

Primul pas al metodei dumneavoastră proprii de înregistrare a jurnalelor este creare tabelei în baza de date. Comanda de creare a unei tabele pe care o puteţi vedea mai jos creează o tabelă denumită `access_tracker` în baza dumneavoastră de date MySQL, având câmpuri pentru un ID, un titlu al paginii, specificatorul `User-agent` al browserul client, precum şi pentru data accesului.

```
mysql> create table access_tracker (
    -> id int not null primary key auto_increment,
    -> page_title varchar(50),
    -> user_agent text,
    -> date_accessed date
    -> );
```

În cele ce urmează, veţi crea un fragment de cod care va scrie aceste lucruri în tabelă.

Crearea unui fragment de cod PHP

După cum aţi dedus deja, fragment de cod înseamnă, în esenţă, o bucăţică mică de cod. Cu alte cuvinte, este vorba despre ceva ce nu poate fi considerat un script lung, dar serveşte un scop anume. În acest caz, fragmentul de cod din listingul 24.1 va scrie câteva informaţii de bază în tabela `access_tracker`.

LISTINGUL 24.1. Fragmentul de cod pentru monitorizarea accesului.

```
1: <?
2: //configureaza variabilele statice
3: $page_title = "sample page A";
4: $user_agent = getenv("HTTP_USER_AGENT");
5:
6: //se conecteaza la server si selecteaza baza de date
7: $conn = mysql_connect("localhost", "joeuser", "somepass") or die(mysql_error());
8: $db = mysql_select_db("testDB", $conn) or die(mysql_error());
```

LISTINGUL 24.1. (*continuare*)

```
 9:
10: //creeaza si lanseaza interogarea
11: $sql = "insert into access_tracker values
12:     ('', '$page_title', '$user_agent',  now())";
13: mysql_query($sql,$conn);
14: ?>
```

Cu acest fragment de cod trebuie să faceţi un lucru foarte simplu: să îl plasaţi la începutul fiecărei pagini pe care vreţi să o urmăriţi. Pentru fiecare pagină, înlocuiţi valoarea variabilei `$page_title` din fragment cu numele efectiv al paginii.

Creaţi un script ilustrativ, denumit `sample1.php`, care să conţină listingul 24.1 şi, în continuare, listingul 24.2.

LISTINGUL 24.2. Un exemplu de pagină HTML.

```
1: <HTML>
2: <HEAD>
3: <TITLE>Sample Page A</TITLE>
4: </HEAD>
5: <BODY>
6: <h1>Sample Page A</h1>
7: <P>Blah blah blah.</p>
8: </BODY>
9: </HTML>
```

Creaţi câteva copii ale acestui fişier, cu diverse nume şi valori ale variabilei `$page_title`. Apoi, accesaţi aceste pagini diferite cu ajutorul browserului dumneavoastră pentru a popula tabela dumneavoastră jurnal.

Crearea unui exemplu de raport

O dată ce aveţi date în tabela dumneavoastră `access_tracker`, puteţi crea o pagină simplă de raportare care să afişeze aceste informaţii. Listingul 24.3 creează un raport care emite interogări SQL pentru a afla numărul total de rezultate, precum şi subtotalurile corespunzătoare fiecărui browser.

LISTINGUL 24.3. Crearea unui raport de acces.

```
1: <?php
2: //se conecteaza la server si selecteaza baza de date
3: $conn = mysql_connect("localhost", "joeuser", "somepass")
4:     or die(mysql_error());
5: $db = mysql_select_db("testDB", $conn) or die(mysql_error());
6:
7: //lanseaza interogarea si selecteaza rezultatele pentru a le numara
8: $count_sql = "select count(page_title) from access_tracker ";
```

LISTINGUL 24.3. (*continuare*)

```
 9: $count_res = mysql_query($count_sql, $conn) or die(mysql_error());
10: $all_count = mysql_result($count_res, 0, "count(page_title)");
11:
12: //lanseaza interogarea si selecteaza rezultatele pentru utilizatori distincti
13: $user_agent_sql = "select distinct user_agent, count(user_agent) as count
14:     from access_tracker group by user_agent order by count desc";
15: $user_agent_res = mysql_query($user_agent_sql, $conn)
16:     or die(mysql_error());
17: //incepe blocul de afisare pentru utilizatori
18: $user_agent_block = "<ul>";
19:
20: //parcurge rezultatele
21: while ($row_ua = mysql_fetch_array($user_agent_res)) {
22:    $user_agent = $row_ua['user_agent'];
23:    $user_agent_count = $row_ua['count'];
24:    $user_agent_block .= "
25:    <li>$user_agent
26:    <ul>
27:    <li><em>accesses per browser: $user_agent_count</em>
28:    </ul>";
29: }
30:
31: //incheie blocul
32: $user_agent_block .= "</ul>";
33:
34: //lanseaza interogarea si selecteaza rezultatele pentru pagini
35: $page_title_sql = "select distinct page_title, count(page_title) as count
36:     from access_tracker group by page_title order by count desc";
37: $page_title_res = mysql_query($page_title_sql, $conn)
38:     or die(mysql_error());
39: //incepe blocul de afisare a titlurilor paginilor
40: $page_title_block = "<ul>";
41:
42: //parcurge rezultatele
43: while ($row_pt = mysql_fetch_array($page_title_res)) {
44:    $page_title = $row_pt['page_title'];
46:    $page_count = $row_pt['count'];
47:    $page_title_block .= "
48:    <li>$page_title
49:        <ul>
50:        <li><em>accesses per page: $page_count</em>
51:        </ul>";
52: }
53:
54: //termina blocul de afisare a titlurilor paginilor
55: $page_title_block .= "</ul>";
56:
57:?>
58: <HTML>
59: <HEAD>
60: <TITLE>Access Report</TITLE>
61: </HEAD>
```

LISTINGUL 24.3. (*continuare*)

```
62: <BODY>
63: <h1>Access Report</h1>
64: <P><strong>Total Accesses Tracked:</strong> <? echo "$all_count"; ?></p>
65: <P><strong>Web Browsers Used:</strong>
66: <?php print "$user_agent_block"; ?>
67: <P><strong>Individual Pages:</strong>
68: <?php print "$page_title_block"; ?>
69: </BODY>
70: </HTML>
```

Liniile 3–5 stabilesc conexiunea la baza de date astfel încât să puteţi emite interogările asupra tabelei `access_tracker`. Liniile 8–10 emit interogarea care selectează numărul total de pagini, iar liniile 13–15 numără înregistrările corespunzătoare tipurilor de browser. Linia 18 deschide o listă neordonată pentru rezultatele interogării asupra specificatorului `user-agent` al browserelor, în vreme ce liniile 21–29 parcurg iterativ rezultatele şi creează lista, care este închisă pe linia 32.

Liniile 35–37 creează şi emit interogarea care numără paginile individuale. Linia 40 deschide o listă neordonată pentru rezultatele acestei interogări, iar liniile 43–52 parcurg iterativ rezultatele şi creează lista paginilor accesate, care este închisă pe linia 55.

Puneţi aceste linii într-un fişier text denumit `accessreport.php` şi amplasaţi acest fişier în directorul rădăcină al serverului dumneavoastră Web. Atunci când accesaţi acest raport, veţi vedea ceva similar figurii 24.1 — numele paginilor dumneavoastră, numerele şi browserele vor fi diferite, dar înţelegeţi ideea.

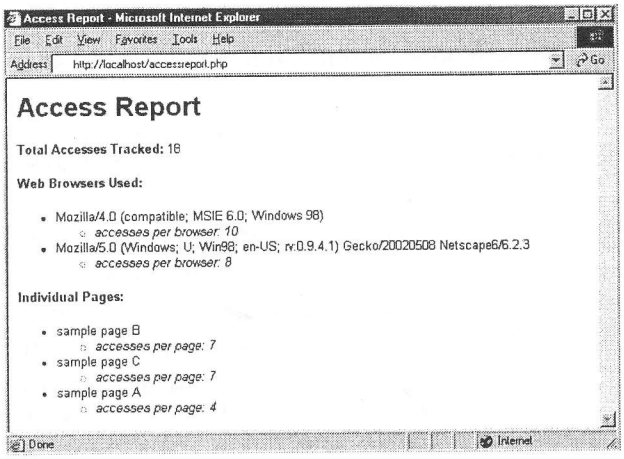


FIGURA 24.1. Un raport de acces personalizat pentru paginile urmărite.

Această modalitate de urmărire este mult mai uşoară decât descâlcirea fişierelor jurnal generate de Apache, dar nu v-aş recomanda să înlocuiţi complet jurnalele de acces cu un sistem orientat pe baze de date. Problema este sarcina suplimentară reprezentată de conexiunile la baza de date, chiar şi în condiţiile în care MySQL este foarte puţin costisitor din punct de vedere al resurselor de sistem. Ce puteţi face, însă, este să limitaţi folosirea sistemului dumneavoastră de urmărire la câteva pagini foarte importante.

Rezumat

Această lecţie v-a explicat cum să înregistraţi în jurnale anumite informaţii despre solicitările primite şi erorile întâlnite de serverul Apache. Puteţi stoca jurnalele în fişiere sau baze de date, sau puteţi să le transmiteţi către aplicaţii externe. Aţi învăţat despre câteva programe utilitare ce vă stau la dispoziţie pentru gestionarea, procesarea şi analizarea jurnalelor, atât despre cele incluse în pachetul Apache, cât şi despre cele create de terţe părţi.

La final, aţi văzut o metodă simplă de folosirea a unui fragment de cod PHP şi a unei tabele MySQL pentru efectuarea unei activităţi elementare de monitorizare a accesului la anumite pagini. Aceste informaţii au fost, apoi, afişate sub forma unu raport de acces simplu, construit în PHP.

Întrebări şi răspunsuri

Î. De ce nu ar trebui să vreau să înregistrez imaginile în jurnal?

R. În cazul serverelor foarte aglomerate, înregistrarea jurnalelor poate să devină o etapă limitatoare de viteză. Dacă scopul jurnalelor este aflarea numărului de vizitatori şi analizarea modului în care aceştia folosesc site-ul, puteţi îndeplini acest scop înregistrând doar accesul la paginile HTML, nu şi la imaginile conţinute de acestea. Astfel, reduceţi numărul de intrări stocate în jurnale şi, implicit, timpul consumat de scrierea acestora.

Atelier

Atelierul este conceput să anticipeze posibile întrebări, să vă ajute să recapitulaţi cele învăţate şi să începeţi să vă puneţi cunoştinţele în practică.

Test

1. Cum aţi evita înregistrarea solicitărilor provenite de la clienţi care accesează site-ul Web folosind o anumită reţea?
2. Cum puteţi să înregistraţi imaginile servite într-un fişier jurnal separat?

Soluţii

1. În unele situaţii, s-ar putea să doriţi să ignoraţi solicitările provenite dinspre o anumită reţea, cum ar fi reţeaua dumneavoastră, astfel încât să nu vă paraziteze rezultatele. Puteţi face acest lucru fie procesând ulterior jurnalele şi eliminând înregistrările respective, fie folosind directiva `SetEnvIf`:

```
SetEnvIf Remote_Addr 10\.0\.0\. intranet
CustomLog logs/access_log "%h %l %u %t \"%r\" %>s %b" !intranet
```

2. Mai devreme, în acest capitol, aţi învăţat cum să evitaţi înregistrarea imaginilor servite în jurnal. În loc să ignoraţi imaginile cu totul, puteţi foarte uşor să le înregistraţi într-un fişier separat, folosind acelaşi mecanism bazat pe variabilele de mediu de operare:

```
SetEnvIf Request_URI "(\.gif|\.jpeg)$" image
CustomLog logs/access_log common env=!image
CustomLog logs/images_log common env=image
```

CAPITOLUL 25
Localizarea aplicațiilor

Cuvintele cheie din sintagma *World Wide Web* sunt *World Wide* (la scară mondială). Folosind PHP şi MySQL este floare la ureche să creaţi un site care să fie util utilizatorilor de naţionalităţi diferite. Procesul de pregătire a aplicaţiilor dumneavoastră pentru folosirea mai multor versiuni locale se numeşte internaţionalizare; personalizarea codului pentru fiecare dintre versiunile locale se numeşte localizare.

În acest capitol, veţi învăţa cum să:

- recunoaşteţi şi vă pregătiţi pentru diferenţele dintre seturile de caractere;
- pregătiţi structura aplicaţiilor dumneavoastră şi să produceţi versiuni localizate.

Despre internaționalizare şi localizare

În primul şi în primul rând, nici internaţionalizarea şi nici localizarea nu sunt echivalente cu traducerea. De fapt, puteţi avea un site Web tradus în întregime — complet în germană, complet în japoneză sau complet în ce limbă doriţi — şi tot nu se va considera că aţi creat un site Web internaţionalizat sau localizat. Veţi avea pur şi simplu un site tradus. Aspectele cheie ale internaţionalizării sunt următoarele:

- externalizarea tuturor şirurilor, pictogramelor şi graficelor;
- modificarea rezultatelor funcţiilor de formatare (pentru datele calendaristice, valorile monetare, numere şi aşa mai departe).

După ce aţi construit aplicaţia dumneavoastră astfel încât toate şirurile de caractere să fie externalizate şi după ce toate funcţiile de formatare au posibilitatea să îşi modifice rezultatele în funcţie de obiceiurile locale, puteţi trece la procesul de localizare. Traducerea reprezintă numai o parte a acestui proces.

O *localizare* este în esenţă o colecţie — în acest caz, o colecţie de şiruri de caractere, grafice, text şi convenţii de formatare care vor fi folosite în site-ul Web ce trebuie localizat. Aceste colecţii sunt de obicei numite pe baza limbii aplicaţiei, cum ar fi, de exemplu, localizarea germană. Deşi pare evident că localizarea germană include text tradus în germană, acest lucru nu înseamnă că site-ul respectiv

nu poate fi folosit decât de către nemţi — austriecii, care vorbesc şi ei tot germana ar putea să folosească un site Web localizat pentru Germania, fără să fie vorbe de o localizare austriacă.

În secţiunile care urmează veţi învăţa să lucraţi cu diferite seturi de caractere ş să modificaţi mediul dumneavoastră de operare, astfel încât să vă puteţi pregăt cu succes aplicaţiile pentru a fi localizate.

Despre seturile de caractere

Seturile de caractere sunt împărţite, de obicei, în două categorii denumite *single byte* (pe un singur octet) şi *multibyte* (multioctet), fapt ce se referă la numărul d octeţi necesari pentru definirea unei relaţii cu un caracter folosit în scrierea une anumite limbi. Engleza, germana şi franceza (printre altele) sunt limbi pe u singur octet; este necesar numai un singur octet pentru a reprezenta un caracte cum ar fi litera *a*, sau numărul 9. Codurile pe un octet au maxim 256 de caractere incluzând întregul set de caractere ASCII, caractere cu accent şi alte caracter necesare pentru formatare.

Codurile multioctet au mai mult de 256 de caractere, incluzând şi toate caracte rele pe un singur octet sub forma unui subset. Limbile multioctet includ chinez tradiţională şi cea simplificată, japoneza, coreeana, thailandeza, araba, ebraic şi aşa mai departe. Aceste limbi au nevoie de mai mult de un octet pentru repre zentarea unui caracter. Un exemplu foarte bun este cuvântul *Tokyo*, capital Japoniei. În engleză, acest cuvânt are patru litere diferite, folosind, în total, 5 octeţ Cu toate astea, în japoneză, cuvântul este reprezentat prin două silabe, *tou kyou*, fiecare folosind câte 2 octeţi, în total 4 octeţi.

Acesta este, probabil, cel mai simplist mod de a explica seturile de caracter şi tehnologia din spatele lor, dar chestiunea relevantă este aceasta: pentru interpreta şi afişa corect textul unei pagini Web într-o anumită limbă, este re ponsabilitatea dumneavoastră să îi spuneţi browserului ce set de caractere s folosească. Acest lucru se realizează prin trimiterea anteturilor corespunzătoa înaintea trimiterii oricăror altor date.

Dacă aveţi un set de pagini care include text în japoneză şi nu trimiteţi ant turile corecte referitoare la limbă şi la setul de caractere, aceste pagini vor afişate incorect într-un browser Web a cărei limbă principală nu este japonez Cu alte cuvinte, pentru că nu este inclusă nici un fel de informaţie privitoa la setul de caractere, browserul presupune că trebuie să afişeze textul folosi propriul său set de caractere implicit. De exemplu, dacă paginile dumneavoast scrise în japoneză folosesc setul de caractere Shift_JIS sau UTF-8, iar browser este configurat să folosească setul de caractere ISO-8859-1, browserul va înc ca să afişeze textul japonez folosind setul de caractere ISO-8859-1, care este un singur octet. Browserul va eşua lamentabil în afara cazului în care dumne voastră îl avertizaţi să folosească Shift_JIS sau UTF-8, iar bibliotecile şi pachete lingvistice corespunzătoare sunt instalate în sistemul de operare.

Anteturile de care vorbim sunt `Content-type` şi `Content-language`, iar ele pot fi folosite şi sub formă de marcaje HTML `META`. Din moment ce aveţi toate instrumentele pentru crearea unui mediu de operare dinamic, cel mai bine este să serviţi anteturile corespunzătoare înainte de trimiterea textului efectiv şi să scrieţi marcajele `META` corespunzătoare în documentele dumneavoastră. În cele ce urmează, veţi vedea câteva exemple de folosire a funcţie `header()` pentru transmiterea informaţiilor corespunzătoare despre limbă şi set de caractere. În cazul unui site în engleză:

```
header("Content-Type: text/html;charset=ISO-8859-1");
header("Content-Language: en");
```

Marcajele `META` asociate ar fi acestea:

```
<META HTTP-EQUIV="Content-Type" content="text/html; charset=ISO-8859-1">
<META HTTP-EQUIV="Content-Language" content="en">
```

Un site în limba germană ar folosi acelaşi set de caractere, dar codul pentru limbă ar fi diferit:

```
header("Content-Type: text/html;charset=ISO-8859-1");
header("Content-Language: de");
```

Marcajele `META` asociate ar fi acestea:

```
<META HTTP-EQUIV="Content-Type" content="text/html; charset=ISO-8859-1">
<META HTTP-EQUIV="Content-Language" content="de">
```

Un site în limba română ar trebui să folosească atât codul de limbă corespunzător, cât şi un set de caractere diferit:

```
header("Content-Type: text/html;charset=ISO-8859-2");
header("Content-Language: ro");
```

Marcajele `META` asociate ar fi acestea:

```
<META HTTP-EQUIV="Content-Type" content="text/html; charset=ISO-8859-2">
<META HTTP-EQUIV="Content-Language" content="ro">
```

Un site în limba japoneză va folosi nu numai un set de caractere diferit, ci şi un cod de limbă propriu:

```
header("Content-Type: text/html;charset=Shift_JIS");
header("Content-Language: ja");
```

Marcajele `META` corespunzătoare ar fi următoarele:

```
<META HTTP-EQUIV="Content-Type" content="text/html; charset=Shift_JIS">
<META HTTP-EQUIV="Content-Language" content="ja">
```

Modificări ale mediului de operare

Mediul dumneavoastră de operare, aşa cum este definit în capitolele referitoare la instalare din această carte, nu trebuie să fie schimbat pentru a putea lucra cu versiuni localizate ale site-urilor Web. Deşi puteţi folosi câteva elemente de configurare legate de limbă în Apache, PHP şi MySQL pentru a uşura lucrul cu site-urile localizate, puteţi efectua toate activităţile din acest capitol fără a face vreo modificare legată de limbă în fişierele dumneavoastră de configurare. Doar

pentru informarea dumneavoastră, în următoarele câteva secţiuni, vi se vor indica sursele de documentare pentru folosirea triadei Apache, PHP şi MySQL în procesele de localizare şi internaţionalizare.

Modificări ale configuraţiei Apache

În capitolul 26, „Reglarea performanţei serverului Apache şi utilizarea host-urilor virtuale", veţi învăţa despre conceptul de negociere a conţinutului prin folosirea modulelor `mod_mime` sau `mod_negotiation`, precum şi, printre altele, a directivelor `AddLanguage` şi `AddCharset`. Aceste directive sun folosite atunci când schimbaţi manual extensiile fişierelor şi vreţi ca Apache să hotărască ce set de caractere să folosească pe baza acestor extensii. În acest capitol, însă, nu discutăm despre acest aspect. Veţi dori ca toate site-urile dumneavoastră localizate să folosească aceleaşi convenţii de denumire a fişierelor (cum ar fi `index.html` sau `info_companie.html`) şi să nu trebuiască să creaţi manual mai multe fişiere cu extensii diferite în funcţie de limba în care este tradus conţinutul. Scopul dumneavoastră în ceea ce priveşte localizarea site-urilor Web este să aveţi un singur set de pagini, servite de către un singur server Web şi care să fie umplute, la momentul execuţiei, cu textul tradus corespunzător.

Apropo

Negocierea conţinutului cu ajutorul modulelor Apache şi folosind mai multe fişiere denumite pe baza unor convenţii de diferenţiere între limbile conţinutului acestora nu reprezintă o greşeală. Capitolul de faţă, însă, nu discută despre această metodă. Puteţi citi mai multe despre negocierea conţinutului cu ajutorul Apache la adresa `http://httpd.apache.org/docs-2Ø/content-negotiation.html`.

Modificări ale configuraţiei PHP

La fel ca şi în cazul serverului Apache, nici configuraţia procesorului de PHP nu trebuie modificată pentru vreuna dintre activităţile din acest capitol. Cu toate acestea, puteţi folosi o grămadă de funcţii legate de manipularea caracterelor multioctet, dacă doriţi să faceţi acest lucru. Aceste funcţii se găsesc în Manualul PHP la adresa `http://www.php.net/mbstring` şi trebuie să fie activate în timpul procesului de configurare prin folosirea codului următor:

```
--enable-mbstring=LANG
```

Aici, `LANG` reprezintă codul limbii, cum ar fi `ja` pentru japoneză, `cn` pentru chineza simplificată şi aşa mai departe. Mai puteţi folosi şi linia de mai jos pentru a activa toate limbile disponibile:

```
--enable-mbstring=all
```

Atunci când funcţiile din categoria `mbstring` sunt activate în PHP, puteţi configura câteva elemente în fişierul `php.ini` pentru ca aceste funcţii să poată fi folosite corect. După ce aţi configurat aceste elemente, puteţi folosi oricare dintre cele mai bine de 40 de funcţii din categoria `mbstring` pentru a manipula caractere multioctet în PHP.

Paginile de manual corespunzătoare acestor funcţii sunt foarte amănunţite şi sunt o lectură obligatorie pentru cei care doresc să efectueze operaţiuni avansate cu seturi de caractere multioctet şi conţinut dinamic. Vă veţi descurca foarte bine cu activităţile din acest capitol fără să le citiţi, dar ar fi recomandabil să răsfoiţi aceste pagini de manual la un moment dat pentru a vă clarifica anumite lucruri.

Modificări ale configurației MySQL

La fel ca şi în cazul Apache şi PHP, nici configuraţia MySQL nu trebuie modificată în vederea folosirii exemplelor de localizare din acest capitol. Setul de caractere implicit folosit de MySQL este ISO-8859-1, dar asta nu înseamnă că nu puteţi stoca în tabelele bazei dumneavoastră de date decât şiruri de caractere pe un singur octet. Eu am o aplicaţie care partajează tabele MySQL codificate în setul de caractere normal ISO-8859-1 şi care conţine texte în engleză, germană, japoneză şi chineză simplificată. Pe de-o parte, acest lucru este din cauză că funcţionalităţile de internaţionalizare şi localizare nu sunt planificate a fi implementate decât în versiunea 4.1 a MySQL (consultaţi capitolul 31, intitulat *Facilităţi noi şi compatibilitatea cu versiunile anterioare ale MySQL 4.1*) şi, pe altă parte, pentru că această variantă funcţionează cât se poate de bine pentru nevoile mele.

Pentru mai multe informaţii despre elementele legate de limbă care se găsesc deocamdată în MySQL, citiţi intrarea din Manualul MySQL la adresa `http://www.mysql.com/en/Localisation.html`.

Crearea structurii unei pagini localizate

În această secţiune, veţi vedea un exemplu funcţional de pagină de întâmpinare care foloseşte PHP pentru a îi permite utilizatorului să selecteze o limbă ţintă şi, apoi, să primească textul corespunzător. Scopul acestei secţiuni este acela de a ilustra modul în care şirurile de caractere folosite în cadrul scripturilor pot fi externalizate, ceea ce reprezintă una dintre caracteristicile fundamentale ale internaţionalizării.

În acest script, se întâmplă ca utilizatorul să ajungă la versiunea în limba engleză a site-ului Web, dar i se oferă posibilitatea de a opta să navigheze folosind versiunea localizată pe care o doreşte — engleză, germană sau japoneză. În acest proces sunt implicate trei elemente:

- crearea şi folosirea unui fişier master care să trimită antetul specific localizării;
- crearea şi folosirea unui fişier master care să afişeze informaţii pe baza localizării selectate;
- folosirea scriptului efectiv.

Listingul 25.1 arată conţinutul fişierului master care este folosit pentru transmiterea antetului cu informaţii specifice pentru localizare.

LISTINGUL 25.1. Fişierul de definire a limbii.

```
1: <?php
2: if ($_SESSION[lang] == "") {
3: $_SESSION[lang] = "en";
4: $currLang = "en";
5: } else {
6: $currLang = $_GET[lang];
7: $_SESSION[lang] = $currLang;
8: }
9: switch($currLang) {
10: case "en":
11: define("CHARSET","ISO-8859-1");
12: define("LANGCODE", "en");
13: break;
14:
15: case "de":
16: define("CHARSET","ISO-8859-1");
17: define("LANGCODE", "de");
18: break;
19:
20: case "ja":
21: define("CHARSET","UTF-8");
22: define("LANGCODE", "ja");
23: break;
24: default:
25: define("CHARSET","ISO-8859-1");
26: define("LANGCODE", "en");
27: break;
28: }
29:
30: header("Content-Type: text/html;charset=".CHARSET);
31: header("Content-Language: ".LANGCODE);
32: ?>
```

Salvaţi acest fişier sub numele `define_lang.php` şi amplasaţi-l în directorul rădăcină al serverului dumneavoastră Web. Acest fişier defineşte două constante ce vor fi folosite în cadrul scriptului următor, care este scriptul de afişare efectiv. Constantele sunt `CHARSET` şi `LANGCODE`, corespunzând setului de caractere şi respectiv, codului limbii pentru fiecare localizare. Aceste constante sunt folosite de către scriptul de afişate pentru a crea marcajele `META` corespunzătoare setului de caractere şi codului limbii. Deşi anteturile sunt trimise de către acest script, o idee bună ar fi să vă asiguraţi că aceste anteturi sunt fac parte şi din pagina efectivă lucru care este folositor pentru datele introduse în eventualele formulare.

Liniile 2–8 ale scriptului din listingul 25.1 stabilesc o valoare de sesiune necesară pentru stocarea limbii selectate de utilizator. Dacă nu există o astfel de valoare de sesiune, sunt folosite valorile corespunzătoare localizării engleze. Dacă site-ul dumneavoastră ar fi fost implicit în germană, ar fi trebuit să schimbaţi acest script

astfel încât să folosească implicit localizarea germană. Acest script pregătește terenul pentru următorul script, care conține un mecanism de selecție, atribuind valoarea variabilei `$currLang` rezultatului acestui mecanism de selecție, pe linia 6.

Instrucțiunea `switch` care începe pe linia 9 conține câteva instrucțiuni `case` concepute pentru a atribui valorile corespunzătoare variabilelor `CHARSET` și `LANGCODE`. Liniile 30–31 chiar folosesc aceste variabile pentru prima oară creând dinamic și trimițând anteturile `Content-type` și `Content-language`.

Listingul 25.2 creează o funcție care pur și simplu stochează șirurile externalizate care vor fi folosite în cadrul scriptului de afișare. Acest exemplu folosește două astfel de șiruri: unul care îi urează bun venit utilizatorului (`WELCOME_TXT`) și altul care prezintă procesul de selecție a limbii (`CHOOSE_TXT`).

LISTINGUL 25.2. Fișierul de definire a șirurilor.

```
1: <?php
2: defineStrings() {
3: switch($_SESSION[lang]) {
4: case "en":
5: define("WELCOME_TXT","Welcome!");
6: define("CHOOSE_TXT","Choose Language");
7: break;
8:
9: case "de":
10: define("WELCOME_TXT","Willkommen!");
11: define("CHOOSE_TXT","Sprache auswählen");
12: break;
13:
14: case "ja":
15: define("WELCOME_TXT","[unprintable characters]");
16: define("CHOOSE_TXT","[unprintable characters]");
17: break;
18:
19: default:
20: define("WELCOME_TXT","Welcome!");
21: define("CHOOSE_TXT","Choose Language");
22: break;
23: }
24: }
25: ?>
```

Folosiți fișierul denumit `lang_strings.php` de pe CD-ul atașat acestei cărți pentru a putea utiliza caracterele japoneze care nu pot fi imprimate aici. Amplasați acest fișier în directorul rădăcină al browserului dumneavoastră Web. Acest fișier definește două constante, `WELCOME_TXT` și `CHOOSE_TXT`, care sunt folosite de către scriptul de afișare. Aceste constante sunt definite în cadrul contextului funcției `defineStrings()`, dar ați putea să scrieți acest fișier sub forma unei instrucțiuni `switch` lungi care să nu se găsească în cadrul vreunei funcții. Am pus totul în interiorul unei funcții de dragul organizării și pentru simplificarea explicațiilor necesare atunci când va veni momentul să folosim scriptul de afișare.

În fine, a venit momentul să creăm scriptul de afişare. Reţineţi că unul dintre elementele cheie ale internaţionalizării este reprezentat de externalizarea tuturor şirurilor de caractere astfel încât numai fişierul master să trebuiască să fie folosit. Listingul 25.3 este un exemplu de astfel de fişier.

LISTINGUL 25.3. Un script de întâmpinare localizat.

```
1: <?
2: session_start();
3: include("define_lang.php");
4: include("lang_strings.php");
5: defineStrings();
6: ?>
7: <HTML>
8: <HEAD>
9: <TITLE><? echo WELCOME_TXT; ?></TITLE>
10: <META HTTP-EQUIV="Content-Type" content="text/html; charset=<? echo CHARSET;
?>">
11: <META HTTP-EQUIV="Content-Language" content="<? echo LANGCODE; ?>">
12: <BODY>
13: <h1 align=center><? echo WELCOME_TXT; ?></h1>
14: <p align=center><strong><? echo CHOOSE_TXT; ?></strong><br><br>
15: <a href="<? echo $_SERVER[PHP_SELF]."?lang=en"; ?>"><img src="en_flag.gif"
border=0></a>
16: <a href="<? echo $_SERVER[PHP_SELF]."?lang=de"; ?>"><img src="de_flag.gif"
border=0></a>
17: <a href="<? echo $_SERVER[PHP_SELF]."?lang=ja"; ?>"><img src="ja_flag.gif"
border=0></a>
18: </p>
19: </BODY>
20: </HTML>
```

Salvaţi acest script sub numele `lang_selector.php` şi amplasaţi-l în directorul rădăcină al serverului dumneavoastră Web. Atunci când este vizitat pentru prima oară, pagina generată de el ar trebui să arate ca în figura 25.1.

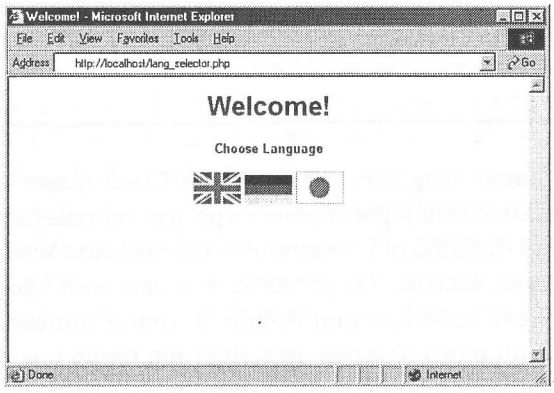


FIGURA 25.1. Vizualizarea selectorului de limbă pentru prima oară.

Până ce este selectată o altă limbă, limba folosită implicit este engleza. Din acest motiv, pe ecran vor apărea textele `Welcome` şi `Choose Language` în engleză.

Uitați-vă la scriptul din listingul 25.3 — este un șablon foarte simplu, deoarece toate elementele legate de limbă sunt externalizate în fișierele `define_lang.php` și `lang_strings.php`. Tot ce face acest al treilea fișier este să afișeze rezultatele corespunzătoare, în funcție de localizarea selectată (sau implicită). Linia 5 apelează funcția `defineStrings()`, care pune la dispoziție valorile corespunzătoare ale celor două constante. Liniile 15–18 afișează steagurile reprezentând localizările engleză, germană și japoneză și pe care se poate da clic. Atunci când utilizatorul dă clic pe unul dintre steaguri, localizarea respectivă va înlocui localizarea curentă, iar șirurile folosite vor fi cele corespunzătoare noii localizări. În aceste linkuri este folosită variabila `lang`, care este pasată scriptului drept `$_GET[lang]`. Dacă vă uitați pe linia 6 a listingului 25.1, veți vedea cum este folosită această variabilă pentru schimbarea valorii referitoare la localizarea selectată de utilizator.

Atunci când utilizatorul dă clic pe steagul Germaniei, el va vedea figura 25.2; când utilizatorul dă clic pe steagul japonez, el va vedea figura 25.3.

FIGURA 25.2. Vizualizarea paginii în limba germană.

FIGURA 25.3. Vizualizarea paginii în limba japoneză.

Folosirea steagului Marii Britanii poate părea oarecum neobişnuită în contextul unei cărţi scrise de un american şi care a fost distribuită iniţial în Statele Unite. Cu toate acestea, când sunt folosite steaguri în loc de nume pentru selectarea localizărilor, steagul Marii Britanii este folosit de obicei pentru reprezentarea localizării engleze, nu cel al Statelor Unite. După cum, probabil, vă imaginaţi, folosirea steagurilor pentru selectarea localizărilor poate să capete încărcătură politică. Companiile care oferă versiuni localizate ale site-urilor lor Web dezbat destul de mult asupra alegerii unei modalităţi de reprezentare pentru selectarea localizărilor — steaguri, nume de ţări, nume de limbi şi aşa mai departe. Nu există o soluţie universală; site-urile companiilor Sun Microsystems şi IBM folosesc nume de ţări, în vreme ce alte companii folosesc nume de limbi, dar până şi Google foloseşte steaguri. Modul în care afişaţi elementele de selecţie a limbii poate reprezenta, fără îndoială, o decizie de afaceri, dar dacă aţi parcurs procesul de externalizare a şirurilor de caractere, a textelor şi a imaginilor creând un şablon internaţionalizat pentru site-ul dumneavoastră Web care este pregătit pentru localizare, formatul elementelor de selecţie a localizării va fi cea mai mică dintre grijile dumneavoastră.

Rezumat

În acest capitol, vi s-au prezentat elementele de bază ale internaţionalizării şi externalizării. Aţi învăţat două lucruri cheie referitoare la crearea unui site internaţionalizat: toate şirurile de caractere, textele şi graficele sunt externalizate şi, la fel, şi formatele de numere, valori monetare şi date calendaristice. Aţi mai învăţat că nici internaţionalizare şi nici localizarea nu sunt echivalente cu traducerea textului; traducerea este numai o parte din procesul de localizare.

Aţi mai învăţat şi câte ceva despre seturile de caractere: acestea pot fi pe un singur octet sau multioctet. Aţi înţeles şi importanţa transmiterii anteturilor corecte referitoare la limbă către browser astfel încât acesta să poată interpreta şi afişa corect textul dumneavoastră.

La final, aţi creat un exemplu practic de stocare a unei variabile de sesiune referitoare la localizare, pentru a stabili şi trimite şiruri de caractere localizate către un şablon preexistent. Acest şablon poate fi folosit de către toate localizările deoarece fiecare element pe care îl conţine este externalizat.

Întrebări şi răspunsuri

Î. *Cum fac să localizez numerele, datele calendaristice şi valorile monetare în PHP?*

R. Două funcţii se vor dovedi extrem de utile în această privinţă: `number_format()` şi `date()`. Aţi învăţat deja să folosiţi funcţia `date()`. Pentru a o folosi într-un mediu de lucru internaţionalizat, trebuie doar să rearanjaţi luna, ziua şi anul astfel încât să se potrivească cu cerinţele localizărilor (`LL-ZZ-AAAA`, `ZZ-LL-AAAA` şi aşa mai departe). Funcţia `number_format()` este folosită pentru numere şi valori monetare; ea separă ordinele de mărime prin virgule, puncte sau spaţii după cum este corect în fiecare localizare. Citiţi intrarea din Manualul de PHP de la adresa `http://www.php.net/number_format` pentru a înţelege cum se foloseşte.

Atelier

Atelierul este conceput să anticipeze posibile întrebări, să vă ajute să recapitulaţi cele învăţate şi să începeţi să vă puneţi cunoştinţele în practică.

Test

1. Limba engleză este o limbă pe un octet sau multioctet? Dar japoneza?
2. Care sunt cele două anteturi referitoare la codificarea caracterelor ce sunt cruciale în contextul unui site localizat?

Soluții

1. Engleza este o limbă pe un singur octet; japoneza este pe doi octeţi.
2. `Content-Type`, incluzând şi indicatorul setului de caractere, şi `Content-Language`.

PARTEA a VI-a

Administrare şi optimizare

Capitolul 26. Reglarea performanței serverului Apache şi folosirea host-urilor virtuale 481

Capitolul 27. Configurarea unui server Web securizat 499

Capitolul 28. Optimizarea şi reglarea serverului MySQL 515

Capitolul 29. Actualizări de software 533

CAPITOLUL 26
Reglarea performanței serverului Apache şi folosirea host-urilor virtuale

În cadrul acestui capitol despre administrare, ne vom concentra asupra creşterii performanţei şi scalabilităţii instalării dumneavoastră de Apache. În plus, veţi învăţa despre folosirea host-urilor virtuale bazate pe nume şi IP, precum şi câteva chestiuni legate de sistemul DNS şi programele client. Vom explica diferite mecanism care pot fi folosite pentru izolarea clienţilor unul de celălalt şi despre avantajele de securitate asociate.

În acest capitol, veţi învăţa:

- ce configurări ale serverului Apache şi ale sistemului de operare pot limita scalabilitatea sau pot degrada performanţa serverului dumneavoastră;
- despre câteva instrumente de testare a încărcării serverului Apache;
- cum să efectuaţi reglajul fin al serverului în vederea obţinerii unei performanţe optime;
- cum să configuraţi Apache pentru a detecta şi preveni comportamente abuzive din partea clienţilor;
- cum să configuraţi host-uri virtuale bazate pe nume şi host-uri virtuale bazate pe IP, precum şi care este diferenţa dintre cele două;
- despre modul în care folosirea host-urilor virtuale depinde de sistemul DNS;
- cum să puneţi la punct matriţe pentru producţia de serie a host-urilor virtuale.

Probleme de scalabilitate

Această secţiune discută despre problemele de scalabilitate şi despre modul în care acestea pot fi prevenite. Mai degrabă este vorba despre o listă de lucruri ce nu trebuie făcute, în cadrul căreia explicăm factorii ce pot degrada performanţa la scalare a serverului. Vom mai investiga şi câteva reglaje ce pot fi efectuate asupra serverului Apache în vederea obţinerii unei performanţe optime.

Limitări ale sistemului de operare

Câţiva factori ai sistemului de operare pot să împiedice scalarea unui server Apache. Aceşti factori ţin de crearea proceselor, de limitele de memorie şi de numărul maxim de fişiere sau conexiuni ce pot fi deschise simultan.

Apropo

Comanda `ulimit` din Unix vă permite să configuraţi câteva dintre limitele discutate în această secţiune pentru fiecare proces în parte. Vă rog să consultaţi documentaţia sistemului dumneavoastră de operare pentru detalii referitoare la sintaxa comenzii `ulimit`.

Procese

Apache oferă câteva configurări pentru prevenirea depăşirii de către server a unui număr maxim de procese şi de fire de execuţie. Aceste configurări afectează scalabilitatea deoarece ele limitează numărul de conexiuni simultane ce pot fi deschise cu serverul dumneavoastră, lucru care, la rândul lui, limitează numărul de vizitatori ce pot utiliza serviciile dumneavoastră în acelaşi timp.

Configurările modulului Apache Multi-Processing Module (MPM) sunt şi ele limitate de configurările sistemului de operare privind numărul de procese şi de fire de execuţie. Metodele prin care aceste limite pot fi schimbate sunt diferite de la un sistem de operare la altul. În cazul kernelurilor Linux 2.0.x şi 2.2.x, trebuie modificată constanta `NR_TASKS` definită în fişierul antet `/usr/src/linux/include/linux/tasks.h` şi, apoi, trebuie recompilat kernelul. În seria de kerneluri 2.4.x, limita poate fi accesată şi la momentul execuţiei prin intermediul fişierului `/proc/sys/kernel/threads-max`. Puteţi citi conţinutul fişierului cu ajutorul comenzii:

```
cat /proc/sys/kernel/threads-max
```

şi puteţi scrie în acest fişier folosind comanda:

```
echo value > /proc/sys/kernel/threads-max
```

În Linux (spre deosebire de majoritatea versiunilor de Unix), există o corelaţie între firele de execuţie şi procese, ele fiind foarte similare din punctul de vedere al sistemului de operare.

În Solaris, aceşti parametri pot fi schimbaţi în fişierul `/etc/system`. Pentru ca modificările aduse acestui fişier să intre în vigoare, nu este necesară recompilarea kernelului, dar sistemul trebuie repornit. Puteţi schimba numărul total de procese schimbând intrarea `max_nprocs` şi puteţi să schimbaţi număr de procese admise pentru un utilizator folosind `maxuproc`.

Descriptorii de fişier

Ori de câte ori un proces deschide un fişier (sau un socket), resursei respective îi este atribuită o structură denumită *descriptor de fişier*, această atribuire încetând în momentul închiderii resursei respective. Sistemul de operare limitează numărul de descriptori de fişier pe care îi poate deschide un proces, limitând astfel numărul de conexiuni simultane pe care le poate primi un server Web. Modul în care aceste

configurări pot fi modificate depinde de sistemul de operare. Pe sistemele Linux, puteți să citiți sau să modificați fișierul `/proc/sys/fs/file-max`. Pe sistemele Solaris, trebuie să editați valoarea intrării `rlim_fd_max` din fișierul `/etc/system`. Această modificare va intra în vigoare doar după repornirea sistemului.

Puteți găsi informații suplimentare despre aceste lucruri la adresa `http://httpd.apache.org/docs/misc/descriptors.html`.

Controlul proceselor externe

Apache ne pune la dispoziție câteva directive de control al resurselor pe care le pot folosi procesele externe. Acest lucru se aplică scripturilor CGI care sunt lansate de către server și programelor executate prin intermediul directivelor `include`, dar nu se aplică scripturilor PHP care sunt invocate atunci când procesorul PHP este instalat ca modul, deoarece modulul rulează în procesul serverului. Următoarele directive Apache pot fi utilizate (în fișierul `httpd.conf`) doar în cazul mediilor de operare de tip Unix și modul lor de folosire diferă de la un sistem la altul:

- **`RLimitCPU`** — Acceptă doi parametri: limita „soft" și limita „hard" pentru timpul de procesor care îi este permis unui proces, exprimate în secunde. Dacă este folosit cuvântul cheie `max`, înseamnă că va fi folosită limita maximă permisă de sistemul de operare. Limita „hard" este opțională. Limita „soft" poate fi schimbată fără repornirea sistemului, iar limita „hard" specifică valoarea maximă pe care o poate lua limita „soft".
- **`RLimitMem`** — Sintaxa acestei directive este identică cu cea a directivei `RLimitCPU`, dar această directivă specifică volumul de memorie ce poate fi folosit de către fiecare proces, exprimată în octeți.
- **`RLimitNProc`** — Sintaxa acestei directive este identică cu cea a directivei `RLimitCPU`, dar această directivă specifică numărul maxim de procese.

Aceste trei directive sunt folositoare atunci când vreți să împiedicați utilizatorii răufăcători sau programele scrise prost să scape de sub control.

Configurări Apache legate de performanță

Această secțiune vă prezintă câteva configurările Apache care afectează performanțele serverului.

Accesul la sistemul de fișiere

Din punct de vedere al resurselor, accesul la fișierele de pe disc este un proces costisitor, așa că ar trebui să reduceți la minim numărul de operațiuni cu discul necesare satisfacerii solicitărilor HTTP.

Legăturile simbolice, fișierele de configurare per-director și negocierea conținutului sunt câțiva dintre factorii care influențează numărul de operațiuni cu discul:

- **Legăturile simbolice** — în Unix, o *legătură simbolică* (*symlink*) este un fişier special care punctează către un alt fişier. Aceste fişiere sunt create cu comanda `ln` şi servesc la a face ca un acelaşi fişier să apară în mai multe locuri.
  Doi dinte parametrii acceptaţi de directiva `Options` sunt `FollowSymLinks` şi `SymLinksIfOwnerMatch`. Apache nu urmează implicit symlink-urile deoarece ar putea duce la ocolirea configurărilor de securitate. De exemplu, puteţi crea un symlink într-o porţiune publică a site-ului Web către un fişier sau director restricţionat, care nu trebuie să fie accesibil pe Web. De aceea, tot implicit, Apache trebuie să execute un test pentru a verifica dacă fişierul solicitat este sau nu un symlink. Dacă este prezent `SymLinksIfOwnerMatch`, Apache va urma un symlink numai dacă utilizatorul care a creat symlinkul are şi statutul de `owner` (proprietar) al fişierului ţintă.
  Din moment ce aceste teste trebuie să fie efectuate pentru fiecare element din fiecare cale care punctează la un obiect din cadrul sistemului de fişiere, ele pot fi costisitoare. În cazul în care aveţi control asupra conţinutului site-ului, trebuie să adăugaţi o directivă `Options +FollowSymLinks` în fişierul de configurare şi să evitaţi argumentul `SymLinksIfOwnerMatch`. În acest fel, testele nu vor mai avea loc, iar performanţele nu va mai fi afectată.

- **Fişiere de configurare per-director** — aşa cum s-a explicat în capitolul 2, intitulat *Instalarea şi configurarea serverului Apache*, este posibil să aveţi fişiere de configurare pentru fiecare director. Aceste fişiere, denumite de obicei `.htaccess`, ne pun la dispoziţie o modalitate convenabilă de configurare a serverului şi permit un oarecare grad de delegare a funcţiilor administrative. Totuşi, dacă această configurare este activată, Apache trebuie să caute aceste fişiere în fiecare dintre directoarele care formează calea către fişierul solicitat, ceea ce duce la un număr semnificativ de operaţiuni suplimentare cu discul. Dacă nu aveţi nevoie de fişiere de configurare pentru fiecare director, puteţi să dezactivaţi această configurare adăugând în fişierul dumneavoastră de configurare directiva `AllowOverride none`. Făcând acest lucru, veţi evita scăderea de performanţă asociată cu accesarea sistemului de fişiere în căutarea fişierelor `.htaccess`.

- **Negocierea conţinutului** — Apache poate servi diferite versiuni ale unui acelaşi fişier pe baza limbii sau a altor preferinţe ale clientului. Acest lucru se bazează pe extensiile fişierelor, dar, la fiecare solicitare primită, Apache trebuie să acceseze sistemul de fişiere de mai multe ori în căutarea fişierelor cu extensia corespunzătoare. Dacă aveţi nevoie să folosiţi negocierea conţinutului, asiguraţi-vă că, cel puţin, folosiţi un fişier cu asocieri de tipuri (type map), fapt ce reduce numărul de operaţiuni cu discul. O metodă alternativă pentru evitarea negocierii conţinutului în scopul internaţionalizării site-urilor poate fi găsită în capitolul 25, intitulat *Localizarea aplicaţiilor*.

- **Fişierul „scoreboard"** — Acesta este un fişier special pe care procesul serverului Apache îl foloseşte pentru a comunica cu procesele sale copil în anumite sisteme de operare mai vechi. Puteţi să specificaţi amplasamentul său cu ajutorul directivei `ScoreBoardFile`, dar atunci când Apache rulează pe platforme moderne nu este nevoie de această directivă. Dacă fişierul este necesar, s-ar putea să creşteţi performanţa amplasându-l pe un disc RAM. Un *disc RAM* este un mecanism care permite ca o porţiune a memoriei să fie accesată ca şi când ar fi un sistem de fişiere. Detaliile privitoare la crearea discurilor RAM diferă de la un sistem la altul.

Configurări privind operațiunile de rețea

Un număr de configurări referitoare la rețea pot degrada performanța serverului Apache:

- **`HostnameLookups`** — atunci când `HostnameLookups` are valoarea `on` sau `double`, Apache va efectua o interogare DNS pentru a captura numele host-ului clientului de fiecare dată când clientul trimite o solicitare. Aceste interogări continue vor crea întârzieri. Valoarea implicită a directivei `HostnameLookups` este `off`. Dacă aveți nevoie de numele host-urilor în fișierele jurnal, puteți să folosiți ulterior un utilitar de rezolvare a numelor de host din jurnale, fără a fi nevoie să faceți acest lucru în timp real.
- **Mecanismul de acceptare** — Apache poate folosi mecanisme diferite pentru a controla modul în care procesele sale copil își însușesc solicitările primite de la clienți. Mecanismul optim depinde de platformă și de numărul de procesoare. Puteți găsi informații suplimentare la adresa `http://httpd.apache.org/docs-2.0/misc/perf-tuning.html`.
- **`mod_status`** — modulul colectează date statistice despre server, conexiuni și solicitări, fapt care încetinește funcționarea serverului. Pentru o performanță optimă, dezactivați-l sau, cel puțin, asigurați-vă că directiva `ExtendedStatus` are valoarea `off`, valoare de altminteri implicită.

Testarea performanțelor serverului cu ajutorul utilitarului ApacheBench

Puteți verifica scalabilitatea și performanța site-ului dumneavoastră cu ajutorul utilitarelor de cuantificare a performanței și de generare de trafic. Există multe utilitare disponibile, atât open source cât și comerciale, fiecare cu diverse grade de complexitate. În general, este foarte dificil de obținut o simulare realistă a traficului deoarece vizitatorii au diferite obiceiuri de navigare, au acces la Internet prin conexiuni cu viteze diferite, întrerup descărcarea fișierelor dacă aceasta durează prea mult, apasă repetat butonul de reîncărcare dacă devin nerăbdători și așa mai departe. Din acest motiv, unele dintre utilitare înregistrează secvențe de trafic real pe care le redau ulterior.

Cu toate acestea, pentru a ne face o imagine rapidă — dar destul de precisă — asupra capacității serverului de gestiona trafic intens, pachetul Apache ne pune la dispoziție un utilitar simplu și util de testare în condiții de încărcare, denumit `ApacheBench`, sau `ab`. Puteți să îl găsiți în directorul `/bin` al distribuției Apache.

Acest utilitar vă permite să solicitați un anumit URL de un număr de ori și să vedeți un rezumat al rezultatelor pe ecran. Următoarea comandă solicită pagina principală a serverului `www.example.com` de 1000 de ori, prin intermediul a 10 clienți virtuali în același timp:

```
# /usr/local/apache2/bin/ab -n 1000 -c 10 http://www.example.com/
```

Apropo

Dacă invocați ab fără nici un argument veți obține o listă completă a opțiunilor și sintaxei sale. În plus, caracterul / din URL-ul țintă este obligatoriu, cu excepția cazului în care este specificată o anumită pagină.

Rezultatele vor arăta asemănător cu cele de mai jos:

```
This is ApacheBench, Version 2.0.40-dev <$Revision: 1.121.2.1 $> apache-2.0
Copyright (c) 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Copyright (c) 1998-2002 The Apache Software Foundation, http://www.apache.org/

Benchmarking www.example.com (be patient)
Completed 100 requests
Completed 200 requests
Completed 300 requests
Completed 400 requests
Completed 500 requests
Completed 600 requests
Completed 700 requests
Completed 800 requests
Completed 900 requests
Finished 1000 requests
Server Software:        Apache/2.0.47
Server Hostname:        www.example.com
Server Port:            80

Document Path:          /
Document Length:        8667 bytes

Concurrency Level:      10
Time taken for tests:   64.525026 seconds
Complete requests:      1000
Failed requests:        0
Write errors:           0
Total transferred:      8911000 bytes
HTML transferred:       8667000 bytes
Requests per second:    15.50 [#/sec] (mean)
Time per request:       0.645 - (mean)
Time per request:       0.065 - (mean, across all concurrent requests)
Transfer rate:          134.86 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:       19   62   59.7 45 727
Processing:   178  572  362.8 478 3151
Waiting:       18  114  176.9 74 1906
Total:        255  634  390.3 536 3301

Percentage of the requests served within a certain time (ms)
  50%    536
  66%    611
  75%    662
  80%    693
  90%    872
  95%   1436
```

```
98%   2162
99%   2461
100%   3301 (longest request)
```

Aceste solicitări au fost făcute prin Internet către un server ilustrativ. Ar trebui să obțineți mult mai multe solicitări pe secundă dacă efectuați testul pe aceeași mașină sau prin rețeaua locală. Rezultatele afișate de acest utilitar sunt cât se poate de clare. Unele dintre cele mai relevante rezultate sunt numărul de solicitări pe secundă și durata medie de răspuns la solicitare. Mai puteți vedea și că mai bine de 90% dintre solicitări au fost satisfăcute în mai puțin de o secundă.

Vă puteți juca cu diferite configurări pentru numărul de solicitări și numărul de clienți paraleli pentru a descoperi punctul de la care serverul dumneavoastră se încetinește considerabil.

Reglarea pozitivă a performanței

Deși secțiunile precedente v-au explicat ce configurări ar putea să împiedice scalarea serverului Apache, în cele ce urmează vi se vor prezenta câteva tehnici de creștere activă a performanței serverului dumneavoastră.

Păstrarea fișierelor în memorie

După cum am explicat mai sus, operațiunile cu discul afectează în mod semnificativ performanța. Deși cele mai multe sisteme de operare moderne păstrează o copie temporară a fișierelor celor mai frecvent accesate în memorie, Apache vă permite să specificați explicit ca anumite fișiere să fie păstrate în memorie astfel încât să nu fie necesare operațiuni cu discul pentru accesarea lor. Modulul care îndeplinește această funcție se numește `mod_file_cache`. Puteți specifica o listă de fișiere care să fie păstrate în memorie folosind directiva MMapFile, care se aplică întregului server. O directivă suplimentară disponibilă în Apache 2.0, `CacheFile`, preia o listă de fișiere, face ca descriptorii lor să fie încărcați în memorie la pornirea serverului și păstrează acești descriptori în memorie între solicitări salvând astfel timp și resurse pentru fișierele solicitate frecvent.

Distribuirea încărcării

O altă cale prin care puteți crește performanța este distribuirea traficului între mai multe servere. Acest lucru poate fi făcut în mai multe feluri:

- prin folosirea unui dispozitiv hardware de echilibrare a încărcării (load balancer) care direcționează traficul de rețea și pe cel HTTP între mai multe servere, din exterior acestea părând să fie un singur server;
- prin distribuirea software a încărcării folosind un proxy inversat împreună cu modulul `mod_rewrite`;

- prin folosirea unor servere separate pentru imagini, fişiere mari şi alte materiale statice. De exemplu, puteţi să puneţi toate imaginile pe un server denumit `images.example.com` şi să puneţi referinţe către ele în paginile HTML stocate pe serverul principal.

Folosirea cache-ului programelor client

Cea mai eficientă modalitate de a servi conţinut este să nu îl serviţi! Acest lucru se poate realiza prin folosirea unor anteturi HTTP corespunzătoare care să instruiască programele client şi proxy-urile referitor la perioada de valabilitate a resurselor solicitate. În acest fel, unele resurse care apar pe mai multe pagini şi care nu se schimbă în mod frecvent, cum ar fi siglele şi butoanele de navigare, sunt transmise numai o singură dată pentru o anumită perioadă de timp.

Pe lângă asta, puteţi folosi modulul `mod_cache` din Apache 2.0 pentru a păstra în memorie pagini generate dinamic astfel încât acestea să nu trebuiască să fie create de fiecare dată. Această metodă ar putea avea un impact enorm asupra performanţei, deoarece paginile generate dinamic necesită, de obicei, accesarea bazelor de date, procesarea şabloanelor şi aşa mai departe, operaţiuni care consumă foarte multe resurse.

Apropo

> La momentul scrierii acestei cărţi, `mod_cache` era încă în fază experimentală. Puteţi citi mai multe despre acest modul la adresa: `http://httpd.apache.org/docs-2.0/mod/mod-cache.html`.

Reducerea volumului de date transmise

O altă cale pentru a reduce încărcarea serverelor este reducerea volumului de date care sunt transmise către clienţi. Drept urmare, site-urile dumneavoastră pot fi accesate mai rapid de către clienţi, în special de către aceia care folosesc conexiuni mai lente. Puteţi face un număr de lucruri în acest scop:

- reducerea numărului de imagini;
- reducerea dimensiunilor imaginilor dumneavoastră;
- comprimarea fişierelor mari ce pot fi descărcate de către clienţi;
- precomprimarea conţinutului HTML static şi negocierea conţinutului;
- folosirea modulului `mod_deflate` pentru a comprima conţinutul HTML. Acest lucru poate fi util dacă aveţi la dispoziţie multă putere de procesare, iar clienţii se conectează prin legături lente. Conţinutul va fi transmis mai rapid, iar procesul se va elibera mai repede pentru a putea procesa noi solicitări.

Configurări de rețea

HTTP 1.1 vă permite să răspundeți la mai multe solicitări prin intermediul unei aceleiași conexiuni. HTTP 1.0 permite același lucru prin intermediul extensiilor `keeep-alive` (de păstrare a conexiunilor inactive). Directiva `KeepAliveTimeout` vă permite să specificați durata maximă, exprimată în secunde, pe care serverul o va aștepta până să închidă o conexiune inactivă. Creșterea duratei de așteptare înseamnă că sporiți șansele ca o conexiune să fie reutilizată. Pe de altă parte, conexiunea respectivă și procesul Apache rămân blocate pe perioada de așteptare, lucru care poate afecta scalabilitatea, așa cum s-a discutat mai sus în acest capitol.

Prevenirea abuzurilor

Atacurile de tip Denial of Service (DoS — refuz al serviciului) funcționează prin înglodarea serverului cu un număr uriaș de solicitări simultane, încetinind serverul sau obligându-l să refuze furnizarea serviciilor solicitate de clienții legitimi. Atacurile DoS sunt greu de prevenit în general și, de obicei, cea mai eficientă abordare este la nivelul rețelei sau al sistemului de operare. Un exemplu de astfel de măsură este blocarea unor anumite adrese astfel încât acestea să nu poată trimite solicitări către server; deși puteți bloca aceste adrese și la nivelul serverului, este mult mai eficient ca ele să fie blocate prin intermediul unor filtre de la nivelul firewall-urilor / router-elor sau instalate în sistemul de operare.

Alte tipuri de abuzuri includ transmiterea unor solicitări extrem de lungi sau deschiderea unui număr foarte mare de conexiuni simultane. Puteți limita dimensiunea solicitărilor și perioada de expirare a acestora. Perioada de expirare implicită a solicitărilor este 300 de secunde, dar puteți să o schimbați folosind directiva `TimeOut`. Un număr de directive vă permit să controlați dimensiunile corpului solicitărilor și ale anteturilor acestora: `LimitRequestBody`, `LimitRequestFields`, `LimitRequestFieldSize`, `LimitRequestLine` și `LimitXMLRequestBody`.

Roboții

Robot, Web spider și *Web crawler* sunt câteva dintre numele pe care le poartă o categorie de programe care accesează paginile site-ului dumneavoastră Web, urmând recursiv toate linkurile din interiorul acestuia. Motoarele de căutare pe Web folosesc astfel de programe pentru a scana Internetul în căutare de servere Web, a descărca conținutul acestor site-uri și a-l indexa. Utilizatorii normali folosesc astfel de programe pentru a descărca porțiuni întregi din site-ul dumneavoastră Web astfel încât să îl poată naviga ulterior, fără a fi conectați la Internet. În mod normal, aceste programe se poartă frumos, dar, uneori, ele pot fi extrem de agresive și pot să supraîncarce site-ul dumneavoastră cu prea multe conexiuni simultane sau se pot bloca în bucle infinite.

Roboţii cuviincioşi vor solicita un fişier special, denumit `robots.txt`, ce conţine instrucţiuni despre cum trebuie accesat site-ul dumneavoastră Web şi despre care părţi ale acestui site nu trebuie considerate disponibile. Sintaxa acestui fişier poate fi găsită la `http://www.robotstxt.org/`. Amplasând un fişier `robots.txt` formatat corespunzător în directorul rădăcină al serverului dumneavoastră Web, puteţi să controlaţi activitatea roboţilor. În plus, puteţi să opriţi solicitările lor la nivelul routerului sau al sistemului de operare.

Implementarea host-urilor virtuale

În vremurile vechi, serverele Web erau concepute pentru a gestiona conţinutul unui singur site. Modalitatea standard pentru găzduirea mai multor site-uri pe aceeaşi maşină consta în instalarea şi configurarea separată a mai multor instanţe ale programului server. Pe măsură ce Internetul a crescut, a crescut şi nevoie de a găzdui mai multe site-uri Web pe un singur calculator şi a fost dezvoltată o soluţie mai eficientă: folosirea host-urilor virtuale. Host-urile virtuale permit ca o singură instanţă de Apache să servească site-uri Web diferite, identificate prin nume de domeniu diferite. Host-urile virtuale *bazate pe IP* au, fiecare, o adresă diferită de IP asociată; host-urile virtuale *bazate pe nume* reprezintă domenii diferite care partajează, toate, aceeaşi adresă de IP.

Programele client pentru Web folosesc sistemul de nume de domeniu (DNS) pentru a traduce numele host-urilor în adrese de IP şi viceversa. Există mai multe tipuri de relaţii posibile:

- **Unul la unul** — înseamnă că fiecare nume de host are o singură adresă de IP unică asociată. Acest lucru se aplică în cazul host-urilor virtuale bazate pe IP.
- **Unul la mai multe** — înseamnă că un singur nume de domeniu este asociat cu mai multe adrese de IP. Acest lucru poate fi util atunci când aveţi mai multe instanţe de Apache care servesc acelaşi site Web. Dacă fiecare dintre servere este instalat pe un calculator diferit, este posibil să echilibraţi traficul Web între aceste calculatoare, îmbunătăţind astfel scalabilitatea.
- **Mai multe la unul** — înseamnă că puteţi aloca o aceeaşi adresă de IP la mai multe nume de host. Clientul va specifica site-ul Web pe care îl accesează folosind antetul `Host:` în cadrul solicitărilor. Acesta este principiul de bază din spatele folosirii host-urilor virtuale bazate pe nume.

Apropo

Atunci când este pusă la punct o asociere de tip mai unul la mai multe, un server DNS poate fi configurat astfel încât să răspundă cu o adresă de IP diferită la fiecare interogare, fapt care poate ajuta la distribuirea încărcării serverelor. Acest tip de rezolvare a numelor de domeniu se numeşte *round robin DNS*. Totuşi, dacă aveţi posibilitatea să utilizaţi un load balancer (dispozitiv hardware de reglare a încărcării) în loc să vă bazaţi pe un server de DNS, puteţi evita problemele legate de relaţia dintre serverul Web şi cel de DNS. Folosind un load balancer, eliminaţi posibilitatea ca traficul intens către serverul dumneavoastră Web să blocheze şi serverul de DNS.

Host-urile virtuale bazate pe IP

Cea mai simplă configurație este atunci când fiecare dintre site-uri are propria adresă de IP unică. Fiecare dintre adresele de IP este asociată cu solicitările HTTP primite de Apache, iar acestea sunt, astfel, tratate separat, folosind zone de date separate definite, fiecare, în containerul său `VirtualHost`, aşa cum se vede mai jos:

```
Listen 192.168.128.10:80
Listen 192.168.129.10:80

<VirtualHost 192.168.128.10:80>
    DocumentRoot /usr/local/www-docs/host1
</VirtualHost>

<VirtualHost 192.168.129.10:80>
    DocumentRoot /usr/local/www-docs/host2
</VirtualHost>
```

Dacă o proprietate `DocumetRoot` nu este specificată pentru un host virtual, configurarea globală, specificată în exteriorul secţiunii `<VirutalHost>`, va fi folosită. În exemplul precedent, fiecare dintre host-urile virtuale are propriul său `DocumentRoot`. Atunci când soseşte o solicitare, Apache foloseşte adresa de IP asociată pentru a direcţiona solicitarea către host-ul corespunzător. De exemplu, dacă soseşte o solicitare destinată adresei de IP `192.168.128.10`, Apache returnează documentele din calea `/usr/local/www-docs/host1`.

Dacă sistemul de operare al maşinii nu poate să rezolve o adresă de IP utilizată în numele unui container `VirtualHost` şi nu există nici o directivă `ServerName`, Apache va anunța la pornire că nu poate să asocieze adresele de IP cu numele de host. Acest anunț nu reprezintă o eroare fatală. Apache va rula în continuare, dar eroarea indică faptul că s-ar putea să mai fie ceva de făcut în ceea ce priveşte configurarea serverului DNS astfel încât browserele să poată găsi serverul dumneavoastră Web. Un nume de domeniu complet calificat (FQDN — *fully qualified domain name*) poate fi folosit în locul adresei IP în numele containerului `VirtualHost` şi în legătura din directiva `Listen` (în cazul în care numele de domeniu se rezolvă prin DNS la o adresă de IP configurată pe maşina pe care rulează serverul şi dacă Apache are posibilitatea să se lege la respectiva adresă de IP).

Host-urile virtuale bazate pe nume

Pentru a reduce consumul de adrese de IP pentru host-urile virtuale, protocolul HTTP/1.1 a introdus antetul `Host:`, care permite browserelor să specifice numele exact al host-ului căruia îi este trimisă solicitarea. Acest lucru permite ca mai multe nume de host să partajeze o aceiaşi adresă de IP. Cele mai multe browsere din zilele noastre oferă suport pentru protocolul HTTP/1.1.

Apropo

> Deşi antetul `Host:` a fost standardizat abia în specificaţia HTTP/1.1, unele browsere mai vechi, care funcţionau pe baza protocolului HTTP/1.0 aveau, totuşi, implementat acest antet.

Un set tipic de anteturi ale unei solicitări emise de Microsoft Internet Explorer este ilustrat în listingul 26.1. În cazul în care URL-ul ar fi fost introdus cu un număr de port, acesta ar fi făcut, de asemenea, parte din conţinutul antetului Host:.

LISTINGUL 26.1. Anteturile solicitării HTTP.

```
GET / HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, */*
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.Ø (compatible; MSIE 5.Ø1; Windows NT 5.Ø)
Host: host1.example.com
Connection: Keep-Alive
```

Apache foloseşte antetul `Host:` pentru a configura modul în care mai multe nume de host pot partaja o aceiaşi adresă de IP — relaţia de tip mai multe la unul de care vorbeam mai sus în acest capitol — şi, de aceea, aceste host-uri virtuale se numesc host-uri virtuale bazate pe nume.

Directiva `NameVirtualHost` vă permite să specificaţi o combinaţie adresă de IP — port prin intermediul căreia serverul va primi solicitări pentru toate host-urile virtuale bazate pe nume. Aceasta este o directivă obligatorie pentru host-urile virtuale bazate pe nume. Listingul 26.2 îi spune serverului Apache să diustribuie toate conexiunile către adresa `192.168.128.1Ø` pe baza anteturilor Host: ale solicitărilor HTTP primite.

LISTINGUL 26.2. Host-uri virtuale bazate pe nume.

```
NameVirtualHost 192.168.128.1Ø
Listen 192.168.128.1Ø:8Ø

<VirtualHost 192.168.128.1Ø>
    ServerName host1.example.com
    DocumentRoot /usr/local/www-docs/host1
</VirtualHost>

<VirtualHost 192.168.128.1Ø>
    ServerName host2.example.com
    DocumentRoot /usr/local/www-docs/host2
</VirtualHost>
```

Pentru fiecare nume de host care rezolvă la adresa de IP `192.168.128.1Ø`, Apache mai poate suporta încă un host virtual bazat pe nume. Dacă soseşte o solicitare către această adresă de IP şi pentru un nume de host care nu este inclus în fişierul de configurare, să zicem `host3.example.com`, Apache va asocia solicitarea cu primul container din fişierul de configurare, în acest caz, `host1.example.com`. Acelaşi comportament se aplică şi solicitărilor care nu sunt

acompaniate de un antet `Host`; oricare dintre containere este primul în fişierul de configurare, acela va fi cel ce primeşte solicitarea.

Un utilizator din interiorul domeniului example.com ar putea să aibă numele de domeniu example.com drept domeniu implicit pentru calculatorul său. În acest caz, el ar putea să îşi direcţioneze browserul către `http://host1/` în loc să folosească numele complet calificat `http://host1.example.com/`. Antetul Host, în acest caz, va conţine doar `host1` în loc de `host1.example.com`. Pentru a vă asigura că solicitarea va fi primită de containerul care trebuie, puteţi folosi directiva `ServerAlias`, aşa cum se poate vedea în listingul 26.3.

LISTINGUL 26.3. Directiva `ServerAlias`.

```
NameVirtualHost 192.168.128.10
Listen 192.168.128.10:80

<VirtualHost 192.168.128.10>
    ServerName host1.example.com
    ServerAlias host1
    DocumentRoot /usr/local/www-docs/host1
</VirtualHost>

<VirtualHost 192.168.128.10>
    ServerName host2.example.com
    ServerAlias host2
    DocumentRoot /usr/local/www-docs/host2
</VirtualHost>
```

De fapt, puteţi să folosiţi în cadrul directivei `ServerAlias` o listă de nume care ar putea apărea în antetul `Host` al solicitărilor, separate prin virgule, astfel încât să nu trebuiască să folosiţi o mulţime de containere `VirtualHost` cu aceleaşi directive pentru a trata toate variaţiunile.

HTTP/1.1 forţează folosirea unui antet `Host`. Dacă versiunea protocolului este identificată ca fiind 1.1 în solicitare, solicitarea *trebuie* să fie acompaniată de un antet `Host`. La începuturile folosirii host-urilor virtuale pe bază de nume, acestea implicau o punerea în balanţă a avantajelor şi dezavantajelor: pe de-o parte, erau necesare mai puţine adrese de IP, dar, pe de altă parte, browserele mai vechi care nu trimiteau anteturi Host şi care încă mai erau folosite nu puteau accesa host-urile virtuale instalate pe server. Astăzi, acest lucru nu mai este valabil deoarece nu există un număr semnificativ statistic de utilizatori care să folosească astfel de browsere vechi.

Host-uri virtuale în serie

În listingurile anterioare, directivele `DocumentRoot` aveau un model simplu:

```
DocumentRoot /usr/local/www-docs/numehost
```

unde *numehost* reprezenta numele host-ului din cadrul numelui complet calificat folosit în directiva `ServerName` a host-ului virtual. Pentru câteva host-uri virtuale, această configuraţie este bună. Dar ce se întâmplă dacă avem nevoie de zeci, sute sau chiar mii de astfel de host-uri virtuale? Fişierul de configurare poate deveni greu de întreţinut. Apache ne oferă o soluţie bună pentru producţia în serie a host-urilor virtuale prin intermediul modulului `mod_vhost_alias`. Puteţi configura serverul Apache astfel încât să asocieze solicitările către host-urile virtuale cu diverse căi din sistemul de fişiere pe baza unor reguli stabilite în directiva `VirtualDocumentRoot`. Această funcţionalitate este în special utilă pentru furnizorii de servicii Internet care vor să pună la dispoziţie câte un host virtual pentru fiecare dintre utilizatorii lor. Următorul exemplu creează o configuraţie simplă pentru host-uri virtuale în masă:

```
NameVirtualHost 192.168.128.10
Listen 192.168.128.10:80

VirtualDocumentRoot /usr/local/www-docs/%1
```

Simbolul `%1` folosit în acest exemplu în directiva `VirtualDocumentRoot` va fi substituit cu prima porţiune a numelui complet calificat. Directivele `mod_vhost_alias` au un limbaj cu ajutorul cărora asociază componentele numelui complet calificat cu căi din cadrul sistemului de fişiere, incluzând caracterele din numele complet calificat.

Dacă am elimina toate containerele `VirtualHost` şi am simplificat configurarea în felul arătat aici, serverul va răspunde solicitărilor pentru fiecare dintre subdirectoarele create în directorul `/usr/local/www-docs`. Dacă porţiunea de nume din cadrul numelui complet calificat se potriveşte cu numele unui astfel de subdirector, Apache va căuta conţinutul acestuia atunci când traduce solicitarea într-o cale din sistemul de fişiere.

Deşi host-urile virtuale moştenesc, de obicei, directivele din contextul serverului principal, unele dintre acestea, cum ar fi directivele Alias, nu se propagă. De exemplu, host-urile virtuale nu vor moşteni această asociere cu sistemul de fişiere:

```
Alias /icons /usr/local/apache2/icons
```

Opţiunea `FollowSymLinks` din cadrul directivei `Options` este, de asemenea, dezactivată în acest context. Cu toate acestea, o variantă a directivei `ScriptAlias` este implementată.

Directiva `VirtualScriptAlias` arătată în fragmentul de mai jos tratează solicitările către orice resurse din directorul `/cgi-bin` care conţine scripturi CGI:

```
NameVirtualHost 192.168.128.10
Listen 192.168.128.10:80
VirtualDocumentRoot /usr/local/vhosts/%1/docs
VirtualScriptAlias /usr/local/vhosts/%1/cgi-bin
```

Reţineţi că `cgi-bin` este un simbol special în cadrul acestei directive; dacă i-am spune directorului doar `cgi`, directiva nu ar funcţiona; trebuie neapărat să fie `cgi-bin`.

Pentru diferite necesităţi legate de host-urile virtuale bazate pe IP, există şi variante ale acestor directive: `VirtualDocumentRootIP` şi `VirtualScriptAliasIP`.

Rezumat

Acest capitol v-a oferit informaţii despre configurările Apache şi ale sistemului de operare care pot afecta scalabilitatea şi performanţa serverului. În cele mai multe cazuri, problemele de scalabilitate ale site-urilor Web sunt legate de generarea conţinutului dinamic şi de accesul la bazele de date. Scrierea unor scripturi eficiente va diminua problemele din aceste categorii. Îmbunătăţirile hardware, cum ar fi plăcile de reţea de înaltă performanţă şi driverele pentru acestea, creşterea memoriei şi bateriile de discuri pot, de asemenea, să contribuie la creşterea performanţei.

În ceea ce priveşte folosirea host-urilor virtuale, Apache poate fi configurat să lucreze cu host-uri virtuale în diverse moduri. Indiferent dacă aveţi nevoie de un număr mare de host-uri turnate pe aceeaşi matriţă sau de un set de host-uri virtuale configurate diferit, sau dacă numărul de adrese de IP pe care le puteţi folosi este limitat, există o cale de a vă configura serverul Apache pentru a vă satisface nevoile. Host-urile virtuale pe bază de nume reprezintă o tehnică uzuală de a implementa host-uri virtuale fără a folosi adrese de IP. Host-urile virtuale pe bază de IP reprezintă o altă metodă pe care o puteţi folosi atunci când aveţi un număr mare de adrese de IP disponibile şi vreţi să păstraţi o configurare elegantă, păstrând un raport de unu la unu între adrese şi host-uri. De asemenea, dacă nu puteţi să vă schimbaţi configuraţia DNS, aveţi posibilitatea de a folosi porturi diferite pentru host-uri diferite.

Întrebări şi răspunsuri

Î. Cum pot să măsor dacă site-ul meu este suficient de rapid?

R. Mulţi dezvoltatori îşi testează site-urile local, sau prin reţeaua locală, dar dacă întreţineţi un site Web public, este foarte probabil ca utilizatorii dumneavoastră să acceseze site-ul prin conexiuni lente. Încercaţi să navigaţi site-ul Web de pe un cont de dialup si asiguraţi-vă că paginile dumneavoastră se încarcă suficient de repede, regula fiind că nici o pagină nu trebuie să se încarce în mai mult de trei secunde.

Î. Cum pot să migrez un host virtual bazat pe nume către propriul său server menţinând, în tot acest timp, continuitatea serviciului?

R. Dacă un host virtual urmează să fie mutat pe o maşină învecinată care, prin definiţie, nu poate avea aceiaşi adresă de IP, puteţi lua câteva măsuri suplimentare. O practică uzuală este să faceţi ceva asemănător cu cele de mai jos, deşi multe variaţiuni sunt posibile:

1. Configuraţi un TTL (*time-to-live* — timp de supravieţuire) cât mai mic asocierilor dumneavoastră DNS. Acest lucru duce la creşterea frecvenţei cu care clienţii de DNS emit interogări pentru numele de domeniu.
2. Configuraţi un alias de IP pe host-ul vechi care să puncteze către noua adresă de IP.
3. Configuraţi conţinutul host-ului virtual astfel încât să fie servit atât printr-un host virtual bazat pe nume cât şi printr-unul bazat pe IP.
4. După ce solicitările către host-ul virtual de la adresa de IP veche încep să se rărească (datorită faptului că în acest moment cache-urile serverelor de DNS încep să expire), serverul dumneavoastră poate fi migrat complet.

Î. Pot să folosesc în acelaşi timp host-uri virtuale bazate pe IP şi pe nume?

R. Da. În cazul în care mai multe adrese de IP sunt legate, puteţi aloca folosirea lor în oricâte feluri. O familie de host-uri virtuale bazate pe nume ar putea fi asociată cu fiecare dintre adresele de IP; nu trebuie decât să folosiţi câte o directivă `NameVirtualHost` diferită pentru fiecare IP. O adresă de IP ar putea fi folosită ca host virtual pe bază de IP pentru SSL, de exemplu, iar o altă adresă de IP ar putea fi dedicată unei familii de host-uri virtuale bazate pe nume.

Atelier

Atelierul este conceput să anticipeze posibile întrebări, să vă ajute să recapitulaţi cele învăţate şi să începeţi să vă puneţi cunoştinţele în practică.

Test

1. Numiţi configurările Apache care ar putea limita scalabilitatea sau ar putea afecta performanţa serverului.
2. Numiţi câteva configurări ale sistemelor de operare care ar putea limita scalabilitatea.
3. Numiţi câteva abordări pentru creşterea performanţei.
4. Care container `VirtualHost` primeşte solicitarea în cazul în care conexiunea foloseşte `NameVirtualHost`, dar nu este trimis nici un antet `Host`?
5. Este obligatorie directiva `ServerName` pentru un container `VirtualHost`?

Soluții

1. Unele dintre configurările Apache care ar putea afecta performanţa şi scalabilitatea sunt argumentele `FollowSymLinks`, `SymLinksIfOwnerMatch` ale directivei `Options`, activarea fişierelor de configurare per director, interogările DNS, utilizarea unui fişier *scoreboard*, şi colectarea de date statistice de către modulul `mod_status`.
2. Unele configurări ale sistemelor de operare care ar putea afecta scalabilitatea includ limitarea numărului de procese, de descriptori de fişier, şi limitarea memoriei alocate fiecărui proces.
3. Următoarele sunt câteva sugestii pentru îmbunătăţirea performanţei: distribuirea încărcării prin intermediul unui load balancer sau a unui proxy inversat, compresia datelor, cachingul, păstrarea fişierelor în memorie şi compilarea statică a modulelor.
4. Citind configuraţia de sus în jos, primul container `VirutalHost` este favorizat. Acelaşi comportament are loc şi dacă există un antet `Host`, dar nici un container `VirtualHost` nu se potriveşte cu conţinutul său.
5. Numai host-urile virtuale bazate pe nume sunt folosite. Antetul Host este comparat cu conţinutul directivei `ServerName`. Dacă nu este satisfăcută condiţia, sunt verificate şi valorile directivelor `ServerAlias` ale containerelor `VirtualHosts`.

CAPITOLUL 27
Configurarea unui server Web securizat

Acest capitol vă explică cum să configuraţi un server Apache capabil de să efectueze tranzacţii în condiţii sigure.

În acest capitol, veţi învăţa despre:

- instalarea şi configurarea modulului Apache `mod_ssl`;
- familia de protocoale SSL/TLS şi conceptele de criptografie pe care se bazează aceste protocoale;
- ce sunt certificatele şi cum pot fi acestea create şi administrate.

Imperativul securităţii

Pe măsură ce Internetul intră în convenţional, iar numărul de companii, persoane şi agenţii guvernamentale care îl folosesc creşte, numărul şi tipul de tranzacţii care necesită un nivel de protecţie creşte şi el. Printre acestea se află tranzacţiile financiare, cum ar fi cazul operaţiunilor bancare şi comerţului electronic, precum şi transmiterea de informaţii cu caracter confidenţial, ca de exemplu fişe medicale sau documente interne ale companiilor. Următoarele nevoi de securitate stau la baza comunicaţiilor sigure pe Internet: confidenţialitate, integritate şi autentificare.

Confidenţialitatea

Confidenţialitatea este cea mai evidentă condiţie a canalelor sigure de comunicare. Dacă dumneavoastră transmiteţi sau accesaţi informaţii cu caracter personal, cum ar fi numărul cardului de credit sau istoricul medical personal, în mod sigur nu v-aţi dori ca vreun străin să intre în posesia acestor informaţii.

Integritatea

Informaţiile conţinute în mesajele transmise trebuie să fie protejate împotriva manipulărilor din exterior. De exemplu, dacă dumneavoastră introduceţi un ordin de cumpărare pentru 100 de acţiuni, nu veţi dori, probabil, ca o altă persoană

să vă intercepteze mesajul şi să îl modifice într-un ordin de achiziţionare pentru 1 000 de acţiuni, sau să înlocuiască mesajul original. În plus, veţi dori să împiedicaţi eventualul atacator să efectueze atacuri de tipul „replay" (reluare), atacuri în care nu se modifică efectiv conţinutul mesajului original, ci acesta este retrimis de câteva ori cu obţinerea unui efect cumulativ.

Autentificarea

Trebuie să decideţi dacă aveţi încredere sau nu în organizaţia sau în persoana cu care comunicaţi. Pentru aceasta, trebuie să autentificaţi identitatea celuilalt participant la conversaţie.

Criptografia este ştiinţa care studiază algoritmii şi metodele folosite pentru transmiterea în siguranţă a mesajelor în scopul asigurării confidenţialităţii, integrităţii şi autenticităţii. Criptoanaliza este ştiinţa decriptării sistemelor de criptare.

Protocolul SSL

SSL este acronimul sintagmei *Secure Sockets Layer*, iar TSL al sintagmei *Transport Layer Security*. Acestea formează o familie de protocoale proiectate iniţial pentru a oferi securitate tranzacţiilor HTTP, dar care pot fi folosite de asemenea pentru o gamă largă de alte protocoale utilizate pe Internet, cum ar fi IMAP şi NNTP. Protocolul HTTP asociat cu SSL poartă numele de *secure HTTP — HTTP securizat*.

Compania Netscape a scos pe piaţă versiunea 2 a protocolului SSL în anul 1994 şi, în 1995, versiunea 3. TLS este un standard IETF creat pentru a standardiza protocolul SSL ca protocol de Internet. TLS este doar o modificare a versiunii 3 de SSL, căruia i s-a adăugat număr mic de facilităţi şi au fost remediate câteva probleme minore. Acronimul TLS este rezultatul unei medieri între companiile Microsoft şi Netscape în ceea ce priveşte numele protocolului, fiecare propunând propriul nume. Totuşi, acest acronim nu s-a impus, majoritatea utilizatorilor referindu-se la protocol sub titulatura de SSL. Cu excepţia cazului în care se specifică explicit altceva, în restul capitolului ne vom referi la aceste protocoale folosind numele SSL. De asemenea, dacă nu se specifică explicit altceva, protocoalele SSL/TLS vor fi denumite *SSL*.

Puteţi indica conectarea la server prin intermediul protocolului SSL înlocuind `http` cu `https` în componenta de protocol a URI-ului. Portul implicit pentru HTTP în conjuncţie cu SSL este 443.

Următoarele secţiuni explică modul în care protocolul SSL asigură cerinţele de confidenţialitate, integritate şi de autentificare enumerate în secţiunea anterioară. În acest proces, vom explica într-o manieră simplificată şi principiile matematice şi de criptografie care stau la baza protocolului SSL.

Asigurarea confidenţialităţii

Protocolul SSL protejează împotriva „tragerii cu urechea" la datele trimise prin criptarea acestora. Criptarea este procesul de convertire a unui mesaj din *text pur*

într-un nou mesaj, criptat, denumit *criptotext*. Deşi textul pur este inteligibil oricărui cititor, criptotextul este complet neinteligibil persoanei care îl interceptează. Decriptarea este procesul invers, care transformă criptotextul în textul pur original.

De obicei, procesele de criptare şi de decriptare implică existenţa unei informaţii suplimentare: o *cheie*. Dacă atât expeditorul cât şi destinatarul posedă aceeaşi cheie, procesul poartă numele de criptografie *simetrică*. Dacă expeditorul şi destinatarul au chei diferite, complementare, procesul se numeşte criptografie *asimetrică*, sau criptografie *cu cheie publică*.

Criptografia simetrică

În cazul în care cheile folosite pentru criptarea, respectiv pentru decriptarea mesajului sunt identice, procesul este cunoscut sub numele de *criptografie simetrică*. DES, Triple-Des, RC4, şi RC2 sunt algoritmi folosiţi în contextul criptografiei cu cheie simetrică. Mulţi dintre aceşti algoritmi au dimensiuni diferite ale cheii, măsurate în biţi. În general, dat fiind un algoritm, un număr mai mare de biţi conţinuţi în cheia sa asigură o securitate mai mare, dar şi o viteză de rulare mai mică, din cauza efortului computaţional mai mare solicitat de algoritmul respectiv.

Criptografia simetrică este relativ rapidă comparativ cu criptografia cu cheie publică, lucru explicat în secţiunea următoare. Criptografia simetrică are însă două dezavantaje principale. Unul dintre ele este necesitatea modificării periodice a cheii, pentru a evita accesul unui „spion" la un număr mare de mesaje criptate cu aceeaşi cheie. Cealaltă problemă este constituită de dificultăţile de distribuţie: cum ajunge cheia de la o persoană la alta într-o manieră sigură? Acesta a fost unul dintre primii factori limitanţi şi, înaintea invenţiei criptografiei cu cheie publică, soluţia era predarea personală a cheii.

Criptografia cu cheie publică

Criptografia cu cheie publică are o abordare diferită. În loc ca cei doi parteneri la conversaţie să folosească o aceeaşi cheie, există o pereche de chei: una publică şi una privată. Cheia publică poate fi distribuită fără probleme, în timp ce proprietarul ţine secretă cheia privată. Aceste două chei sunt complementare — un mesaj criptat cu una dintre chei nu poate fi decriptat decât cu cealaltă cheie.

Oricine doreşte să vă trimită un mesaj securizat poate cripta mesajul folosind cheia dumneavoastră publică, fiind sigur că numai proprietarul cheii particulare — dumneavoastră — puteţi să îl decriptaţi. Chiar dacă atacatorul are acces la cheia publică, el nu poate decripta comunicarea. De fapt, dumneavoastră chiar vă doriţi ca această cheie publică să fie cât mai disponibilă cu putinţă. Criptografia cu cheie publică poate fi folosită, de asemenea, în sensul asigurării integrităţii mesajului şi a autentificării. RSA este cel mai popular algoritm de criptare cu cheie publică.

Persoanele care deţin chei publice le vor plasa pe servere de chei publice, sau le vor trimite pur şi simplu corespondenţilor cu care doresc să întreţină conversaţii e-mail securizate. Folosind instrumentele corespunzătoare, ca de exemplu PGP sau GnuPG, expeditorul va cripta mesajele trimise folosind cheiea publică a destinatarului.

Afirmaţia că mesajul poate fi decriptat numai de către deţinătorul cheii particulare înseamnă doar că, în condiţiile actuale ale cunoştinţelor despre criptografie şi a disponibilităţii de putere de calcul, atacatorul nu va fi capabil să „spargă" criptarea exclusiv prin forţă brută, într-un interval de timp rezonabil. Dacă algoritmul sau implementarea sistemului de criptare sunt însă deficitare, există posibilitatea rezonabilă a atacurilor.

Apropo

Criptografia cu cheie publică este ca şi când aţi pune la dispoziţie un număr mare de lacăte identice, păstrâng cheia care le deschide pe toate. Oricine doreşte să vă trimită un mesaj în condiţii de securitate poate să o facă plasând mesajul într-un seif şi încuind seiful cu unul dintre lacătele (cheile publice) oferite de dumneavoastră înainte de a-l trimite. Numai dumneavoastră aveţi cheia corespunzătoare (cheia privată) care va deschide lacătul respectiv (va decripta mesajul).

Protocolul SSL foloseşte criptografia cu cheie publică într-o fază iniţială de *handshake* (crearea canalului de comunicare) pentru a transmite în condiţii de securitate cheile simetrice ce sunt utilizate ulterior pentru decriptarea comunicării.

Asigurarea integrităţii

Efectuarea unui calcul special asupra conţinutului mesajului şi stocarea rezultatului împreună cu conţinutul efectiv al mesajului poate ajuta la conservarea integrităţii datelor. Când mesajul ajunge la destinaţie, destinatarul poate efectua acelaşi calcul, comparând ulterior rezultatele. Dacă s-ar fi efectuat modificări asupra conţinutului mesajului, rezultatele calculului ar fi diferite.

Algoritmii de „digerare" (digest) urmează exact acest proces, creând rezumate ale mesajelor. Un astfel de rezumat (*message digest*) este o metodă de creare a unei reprezentări de lungime fixă a unui mesaj oarecare, care îl identifică în mod unic. Puteţi să îl priviţi ca o amprentă a mesajului. Un algoritm corespunzător de digest ar trebui să fie ireversibil şi rezistent la coliziuni, cel puţin în ceea ce priveşte scopurile practice. *Ireversibil* înseamnă că din mesajul rezumat nu poate fi obţinut mesajul original, iar *rezistent la coliziuni* înseamnă că două mesaje diferite nu pot să aibă acelaşi rezumat asociat. Exemple clasice de algoritmi de digest sunt MD5 şi SHA.

Mesajele rezumate, însă, nu garantează de unele singure integritatea mesajului, deoarece un atacator ar putea modifica textul *şi* mesajul rezumat. Codurile de autentificare de mesaj (*message autentification code* — MAC) sunt similare mesajelor rezumate, dar încorporează în proces o cheie comună secretă. Rezultatul

algoritmului depinde atât de mesaj, cât şi de cheia utilizată. Întrucât atacatorul nu are acces la cheie, el nu poate modifica atât mesajul cât şi rezultatul digerării acestuia. HMAC este un exemplu de algoritm de autentificare de mesaje.

Protocolul SSL foloseşte codurile MAC pentru a evita atacurile de tip reluare, replay, şi pentru a asigura integritatea informaţiilor transmise.

Asigurarea autenticităţii

Protocolul SSL foloseşte certificate pentru a autentifica participanţii la o conversaţie. Criptografia cu cheie publică poate fi folosită pentru semnarea digitală a mesajelor. De fapt, prin simpla criptare a mesajului cu cheia dumneavoastră secretă, destinatarul poate să fie sigur că mesajul provine într-adevăr de la dumneavoastră. Alţi algoritmi de semnătură digitală implică iniţial calcularea unui mesaj rezumat, urmată de semnarea mesajului respectiv.

Puteţi fi sigur că persoana care a creat perechea cheie publică/cheie personală este într-adevăr persoana care trimite mesajul. Cum puteţi însă să asociaţi cheia respectivă cu o persoană sau o organizaţie în care aţi avea încredere în lumea reală? Teoretic, un atacator ar putea pretinde o identitate falsă, urmând să distribuie o cheie publică diferită şi pretinzând că aceasta este cheia reală. Încrederea este în aceste condiţii câştigată cu ajutorul certificatelor digitale. *Certificatele digitale* sunt documente electronice care conţin o cheie publică şi informaţii despre posesorul acesteia (nume, adresă etc.). Pentru a fi util, acest certificat trebuie să fie semnat de o terţă parte de încredere (*autoritatea de certificare* — AC) care verifică corectitudinea informaţiilor respective. Sunt mai multe tipuri de autorităţi de certificare, aşa cum veţi afla pe parcursul acestui capitol. Unele dintre ele sunt entităţi comerciale, care oferă servicii de certificare companiilor ce desfăşoară afaceri pe Internet. Există şi companii care pun la dispoziţie servicii interne de certificare, care constituie un alt grup de AC.

AC garantează că informaţiile din certificat sunt corecte, cheia aparţinând într-adevăr persoanei sau organizaţiei respective. Certificatele au o perioadă de valabilitate şi pot expira sau pot fi revocate. Certificatele pot fi arborescente, astfel încât procesul de certificare să poată fi delegat. De exemplu, o entitate de certificare poate emite certificate companiilor, iar acestea, lar rândul lor, pot realiza certificarea propriilor angajaţi.

Pentru ca acest proces să fie eficient şi demn de încredere, autoritatea de certificare trebuie să solicite dovezi corespunzătoare ale identităţii persoanelor şi organizaţiilor înainte de a le elibera un certificat.

Browserele conţin implicit o colecţie de certificate „de nivel zero" aparţinând unor autorităţi de certificare de renume.

Protocolul SSL şi certificatele

Cele mai standardizate certificate sunt X.509, adaptate utilizării pe Internet. Un certificat X.509 conţine următoarele informaţii:

- **Emiţător** — numele persoanei care semnează certificatul;
- **Subiect** — persoana care deţine cheia ce a fost certificată;
- **Cheia publică a subiectului;**
- **Informaţii de control** — date, cum ar fi perioada de valabilitate a certificatului;
- **Semnătura** — semnătura care acoperă datele anterioare.

Puteţi să studiaţi un certificat real conectându-vă browserul la un server securizat. În cazul în care s-a reuşit conectarea, o pictogramă reprezentând un lacăt sau un alt simbol vizual va apărea în bara de stare a browserului dumneavoastră. În funcţie de browserul pe care îl folosiţi, ar trebui să puteţi da clic pe iconiţa respectivă pentru a vedea informaţii despre conexiunea SSL şi despre certificarea serverului la care v-aţi conectat. Deschideţi URL-ul `https://www.zend.com/` în browser şi analizaţi-i certificatul, căutând elementele prezentate în paragraful anterior. Puteţi observa că emiţătorul certificatului este Thawte CA. Pagina a fost încărcată fără intervenţia dumneavoastră, deoarece Thawte este o autoritate de certificare de prestigiu, ale cărei certificate fac parte din pachetul de pornire al browserelor Internet Explorer şi Netscape Navigator.

Puteţi observa că atât emiţătorul cât şi subiectul sunt desemnaţi cu nume distincte (*distinguished names* — DN), ceea ce constituie o modalitate de oferire a unui identificator unic pentru fiecare element din reţea. În cazul certificatului Thawte, numele distinct este C=IL, S=Mehoz Tel Aviv, L=Ramat Gan, O=Zend Technologies, Ltd., CN=`www.zend.com`.

C este iniţiala de la *country* (*ţară*), S iniţiala de la *state* (*stat*), L de la *locality* (*localitate*), O iniţiala pentru *organization* (*organizaţie*), iar CN reprezintă *common name* (*numele comun*). În cazul certificatului unui site Web, numele comun identifică numele de domeniul complet calificat al siteului Web. Aceasta este partea din URL corespunzătoare numelui serverului; în cazul nostru este vorba de `www.zend.com`. Dacă această porţiune nu corespunde adresei introduse de dumneavoastră în bara de adrese, browserul va genera un mesaj de eroare.

Rezumatul protocolului SSL

Aţi aflat până în acest moment cum asigură protocolul SSL confidenţialitatea prin intermediul criptării, integritatea cu ajutorul codurilor de autentificare, şi autenticitatea folosind certificate şi semnături digitale.

Procesul de stabilire a unei conexiuni SSL este următorul:

1. Utilizatorul foloseşte browserul pentru a se conecta la serverul Apache.
2. Începe faza iniţială, handshake (de „strângeri de mână") — browserul şi serverul fac schimb de chei şi de informaţii de certificare.
3. Browserul verifică valabilitatea certificatului serverului, inclusiv data de expirare, dacă a fost emis de o AC de încredere, etc.
4. Opţional, serverul poate solicita clientului prezentarea unui certificat, de asemenea valid.
5. Serverul şi clientul folosesc fiecare cheia publică a celuilalt pentru a ajunge la un consens despre o cheie simetrică.
6. Se termină faza de handshake, iar transmisia continuă folosind criptografia simetrică.

Obţinerea şi instalarea instrumentelor SSL

Suportul SSL este oferit de modulul Apache `mod_ssl`. Acest modul necesită prezenţa bibliotecii `OpenSSL` — o implementare de tip open source a protocoalelor SSL/TLS, împreună cu o gamă largă de algoritmi de criptare suplimentari. OpenSSL se bazează pe biblioteca `SSLeay`, dezvoltată de Eric A. Young şi Tim J. Hudson.

Din cauza restricţiilor de distribuţie a tehnicilor de criptografie a şirurilor şi a patentelor aplicabile mondial, instalarea instrumentelor conexe SSL poate fi un proces mai mult sau mai puţin complex, în funcţie de platformă. Secţiunile următoare vă prezintă o trecere în revistă a obţinerii şi instalării instrumentelor conexe protocolului SSL.

Biblioteca OpenSSL

Toate fişierele şi instrucţiunile necesare pentru instalarea bibliotecii OpenSSL sunt disponibile la adresa `http://www.openssl.org/`. Utilizatorii de Linux/Unix (şi de diferite variante ale acestora) vor observa că instalarea aplicaţiei software OpenSSL este foarte similară instalării altor utilitare de sistem. Utilizatorul obişnuit al sistemului de operare Windows va observa, însă, că în momentul de faţă nu există nici o distribuţie pre-compilată de fişiere binare. Ca urmare, utilizatorii de Windows trebuie să compileze singuri librăria OpenSSL.

O dată ce aţi instalat pachetul OpenSSL, veţi avea toate elementele necesare pentru crearea şi manipularea certificatelor şi cheilor, precum şi pentru interfaţarea modulului Apache `mod_ssl`.

Pentru utilizatorii de Windows

Utilizatorii sistemului de operare Windows familiarizaţi cu procesul de compilare a propriilor fişiere binare pot să folosească astfel codul sursă OpenSSL oferit pe site-ul Web OpenSSL. Instrucţiunile pentru compilarea OpenSSL în Windows se află în fişierul `INSTALL.W32`, aflat în distribuţia surselor. Prezentarea acestor instrucţiuni depăşeşte scopul cărţii noastre, iar dumneavoastră veţi descoperi că ele sunt complete, inteligibile şi bine scrise. Instrumentele necesare sunt ActiveState Perl pentru Windows şi unul dintre următoarele compilatoare pentru limbajul C:

- Visual C++;
- Borland C;
- GNU C (Cygwin sau MinGW).

Asiguraţi-vă că urmaţi instrucţiunile corespunzătoare compilatorului dumneavoastră preferat, deoarece ele diferă destul de mult de la un compilator la altul. Puteţi obţine, de asemenea, sfaturi despre compilarea OpenSSL de la Apache, la adresa `http://httpd.apache.org/docs-2.0/platform/win_compiling.html`.

Pentru utilizatorii de Linux/Unix

În cazul în care rulaţi o versiune recentă a unei distribuţii de Linux sau FreeBSD, s-ar putea ca librăria OpenSSL să fie deja instalată pe calculatorul dumneavoastră. În cazul în care trebuie, însă, să instalaţi OpenSSL, puteţi descărca codul sursă de pe site-ul OpenSSL. O dată descărcat acest cod, decomprimaţi-l şi executaţi comanda `cd` în directorul creat (înlocuiţi termenul general `-versiune` din comenzile următoare cu versiunea dumneavoastră, actualizată, de OpenSSL):

```
# gunzip < openssl-versiune.tar.gz | tar xvf -
# cd openssl-versiune
```

Instrucţiunile complete de instalare se găsesc în fişierul `INSTALL`, dar, pe scurt, scriptul `config` vă va ajuta să construiţi aplicaţia software, urmat de procesele `make` şi `make install`.

Modulul `mod_ssl`

În trecut, extensiile SSL pentru Apache trebuia să fie distribuite separat, din cauza restricţiilor de export. În prezent, modulul `mod_ssl` este livrat împreună cu Apache 2.0, dar numai drept componentă a distribuţiei surselor. În timp ce aceasta nu este o problemă pentru utilizatorii de Linux/Unix, utilizatorii de Windows se vor confrunta cu necesitatea compilării surselor serverului Apache pentru a putea utiliza modulul `mod_ssl`; acest modul nu face parte din fişierele binare pre-compilate şi distribuite. Modulul `mod_ssl` este dependent de biblioteca OpenSSL, care trebuie să fie instalată pentru ca acesta să poată fi compilat.

Pentru utilizatorii de Windows

Pentru a putea utiliza modulul `mod_ssl`, trebuie să compilaţi pachetul Apache pornind de la zero. Cu alte cuvinte, dacă aţi urmat instrucţiunile de instalare din capitolul 2, intitulat *Instalarea şi configurarea serverului Apache*, puneţi-le deoparte şi respectaţi documentaţia Apache, pe care o găsiţi la adresa `http://httpd.apache.org/docs-2.0/platform/win_compiling.html`.

Din nou, prezentarea instrucţiunilor respective nu intră în scopul acestei cărţi, dar noi vă promitem că ele vă vor oferi toate informaţiile de care aveţi nevoie. Condiţiile esenţiale sunt:

- pachetul OpenSSL instalat;
- Microsoft Visual C++ 5.0 sau o versiune ulterioară;
- Windows Platform SDK;
- utilitarul awk (awk, gawk, sau un instrument similar).

Pentru utilizatorii de Linux/Unix

Distribuţia de surse folosită în capitolul 2 ar trebui să includă deja fişierele necesare pentru folosirea `mod_ssl`. În această situaţie, pentru a utiliza `mod_ssl` trebuie numai să rulaţi din nou procesele `configure` şi `make install`, adăugând următoarele ca parte a comenzii `configure`:

```
--enable-ssl --with-ssl=/usr/local/ssl/
```

În linia de mai sus presupunem că aţi instalat librăria OpenSSL în directorul sugerat de noi; dacă ea se află în alt director de pe serverul dumneavoastră este suficient să înlocuiţi calea cu cea corectă.

Dacă aţi compilat modulul `mod_ssl` static în Apache, puteţi să verificaţi dacă este prezent cu ajutorul următoarei comenzi, care vă oferă o lista de module compilate:

```
# /usr/local/apache2/bin/httpd -l
```

Comanda de mai sus presupune că aţi instalat serverul Apache în directorul `/usr/local/apache2`.

În cazul în care modulul `mod_ssl` a fost compilat ca un modul încărcat dinamic trebuie să adăugaţi sau să îndepărtaţi caracterul de comentariu (#) din faţa următoarei linii de cod din fişierul de configurare:

```
LoadModule ssl_module modules/libmodssl.so
```

Administrarea certificatelor

După ce aţi instalat şi aţi configurat biblioteca OpenSSL şi modulul `mod_ssl`, următorul pas spre o implementare reuşită a protocolului SSL este crearea unui certificat de server. Această secţiune vă va explica în detaliu cum să creaţi şi să administraţi certificate cu ajutorul utilitarului din linie de comandă `openssl`. De exemplu, dacă folosiţi un site SSL de e-comerţ, criptarea le oferă clienţilor protecţie în faţa eventualilor „spioni", iar certificatul le permite clienţilor să verifice dacă dumneavoastră sunteţi într-adevăr cine pretindeţi a fi.

Apropo

Exemplele se referă la versiunea Unix a programului din linie de comandă `openssl`. Dacă dumneavoastră folosiţi sistemul de operare Windows, trebuie să apelaţi în schimb executabilul `openssl.exe` şi să modificaţi calea din exemplele prezentate, folosind de asemenea caractere backslash (\) unde este cazul. Dacă instalarea dumneavoastră de OpenSSL se află într-un director diferit de cel utilizat de noi în secvenţele de mai jos, este suficient să înlocuiţi expresia cu calea corectă.

Crearea unei perechi de chei

Trebuie să deţineţi o pereche cheie publică/cheie privată înainte de a putea crea o solicitare de certificat. Să presupunem că elementul FQDN pentru certificatul pe care doriţi să-l creaţi este `www.example.com`. (Va trebui să înlocuiţi acest FQDN cu elementul FQDN corespunzător maşinii pe care rulează serverul dumneavoastră Apache.) Puteţi crea această pereche de chei cu ajutorul următoarei comenzi:

```
# /usr/local/ssl/bin/openssl genrsa -des3 -rand file1:file2:file3 \
        -out www.example.com.key 1024
```

Parametrul `genrsa` anunţă librăria OpenSSL că doriţi să generaţi o pereche de chei.

Parametrul `des3` indică o solicitare de cheie privată criptată şi protejată printr-o frază tip parolă.

Parametrul `rand` este folosit pentru a oferi bibliotecii OpenSSL numere aleatorii, care să asigure unicitatea şi lipsa de predictabilitate a cheilor generate. În acest scop, substituiţi parametrii `file1`, `file2`, etcetera cu calea către câteva fişiere mari şi cât mai diferite (o imagine de kernel, fişiere jurnal comprimate, şi aşa mai departe). Puteţi să folosiţi deasemenea calea `/dev/random`, dacă aceasta există pe sistemul dumneavoastră. Parametrul `rand` nu este necesar în cazul unui sistem Windows, deoarece numerele aleatorii sunt generate în acest caz automat, prin alte metode.

Parametrul `out` indică locul în care trebuie stocat rezultatul.

Parametrul `1024` se referă la numărul de biţi al cheii generate.

Rezultatul apelării acestei comenzi arată în modul următor:

```
625152  semi-random bytes loaded
Generating RSA private key, 1024 bit long modulus
.....++++++
.........................++++++
e is 65537 (0x10001)
Enter PEM pass phrase:
Verifying password - Enter PEM pass phrase:
```

După cum puteţi observa, vi se va solicita să introduceţi o parolă; alegeţi o frază sigură. Parola este necesară pentru protejarea cheii particulare, şi va trebui introdusă la fiecare pornire a serverului. Puteţi alege să nu protejaţi cheia; acest lucru poate fi convenabil, deoarece nu va mai fi nevoie să introduceţi o parolă la repornirea calculatorului, dar o asemenea practică este total lipsită de securitate, şi comprimiterea protecţiei oferite de server induce de asemenea compromiterea protecţiei oferită de sistemul de chei. În orice caz, puteţi selecta să nu protejaţi cheia, fie eliminând parametrul `-des3` în faza de generare, fie folosind comanda următoare:

```
# /usr/local/ssl/bin/openssl rsa -in www.example.com.key \
        -out www.example.com.key.unsecure
```

O idee bună este să faceţi o copie a fişierului `www.example.com.key`. Puteţi studia conţinutul acestui fişier cu ajutorul următoarei comenzi:

```
# /usr/local/ssl/bin/openssl rsa -noout -text -in www.example.com.key
```

Crearea unei solicitări de semnare a certificatului

Pentru a obţine un certificat semnat de o AC, trebuie să trimiteţi ceea ce se numeşte o *solicitare de semnare de certificat* (*certificate signing request*). Pentru a crea o asemenea cerere, rulaţi comanda următoare:

```
# /usr/local/ssl/bin/openssl req -new -key www.example.com.key \
        -out www.example.com.csr
```

Vă vor fi solicitate următoarele informaţii despre certificat:

```
Using configuration from /usr/local/ssl/install/openssl/openssl.cnf
Enter PEM pass phrase:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:US
State or Province Name (full name) [Some-State]:CA
Locality Name (eg, city) []: San Francisco
Organization Name (eg, company) [Internet Widgits Pty Ltd]:.
```

```
Organizational Unit Name (eg, section) []:.
Common Name (eg, YOUR name) []:www.example.com
Email Address []:administrator@example.com
Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

Este foarte important ca intrarea din câmpul `Common Name` să corespundă adresei pe care vizitatorii site-ului dumneavoastră Web o vor tasta în bara de adrese a browserului. Aceasta este una dintre verificările pe care browserul le efectuează asupra certificatului serverului apelat. Dacă numele diferă, utilizatorul va primi un mesaj de avertizare în care este prezentată problema.

În acest moment, certificatul este stocat în `www.example.com.csr`. Puteţi afla mai multe despre conţinutul certificatului cu ajutorul următoarei comenzi:

```
# /usr/local/ssl/bin/openssl req -noout -text \
        -in www.example.com.csr
```

Puteţi trimite în continuare fişierul cu solicitarea de semnare ce certificat către AC pentru procesare. Două dintre cele mai prestigioase autorităţi de certificare sunt VeriSign şi Thawte. Puteţi afla mai multe despre procedurile de trimitere specifice de pe site-urile lor Web:

- **VeriSign** — `http://www.verisign.com/products/site/index.html`
- **Thawte** — `http://www.thawte.com/html/RETAIL/ssl/index.html`

Crearea unui certificat autosemnat

Puteţi crea, de asemenea, un certificat autosemnat; cu alte cuvinte, dumneavoastră veţi fi atât emiţătorul, cât şi subiectul certificatului. Deşi acest lucru nu este prea poate de util în cazul unui site Web comercial, vă va permite să testaţi instalarea modulului `mod_ssl`, precum şi să rulaţi un server Web securizat în timp ce aşteptaţi eliberarea certificatului oficial de către AC.

```
# /usr/local/ssl/bin/openssl x509 -req -days 30  \
        -in www.example.com.csr -signkey www.example.com.key \
        -out www.example.com.cert
```

Va trebui să copiaţi certificatul dumneavoastră `www.example.com.cert` (fie cel primit de la AC, fie cel autosemnat) la adresa `/usr/local/ssl/openssl/certs/` şi cheia dumneavoastră la adresa to `/usr/local/ssl/openssl/private/`.

Protejaţi-vă fişierul care conţine cheia privată cu ajutorul următoarei comenzi:

```
# chmod 400 www.example.com.key
```

Configurarea protocolului SSL

Secţiunile anterioare v-au prezentat conceptele (nu atât de simple) din spatele protocolului SSL, iar dumneavoastră aţi învăţat cum să generaţi chei şi certificate. În acest moment puteţi să configuraţi serverul Apache pentru a accepta protocolul SSL. Modulul `mod_ssl` trebuie fie compilat static, fie, dacă l-aţi compilat sub formă de modul încărcat dinamic, să introduceţi în fişierul de configurare directiva `LoadModule` corespunzătoare.

În continuare, puteţi adăuga în fişierul de configurare Apache următorul fragment de cod:

```
Listen 443
<VirtualHost _default_:443>
ServerName www.example.com
SSLEngine on
SSLCertificateFile \
/usr/local/ssl/openssl/certs/www.example.com.cert
SSLCertificateKeyFile \
/usr/loca/ssl/openssl/certs/www.example.com.key
</VirtualHost>
```

Această porţiune de cod configurează un host virtual nou care va asculta pe portul 443 (portul implicit pentru HTTPS), iar dumneavoastră activaţi protocolul SSL în hostul virtual respectiv prin directiva `SSLEngine`. Directivele `SSLCertificateFile` şi `SSLCertificateKeyfile` indică amplasamentul certificatului serverului şi al fişierului ce conţine cheia asociată.

Pornirea serverului

Puteţi opri serverul dacă acesta rulează, urmând să-l porniţi din nou. Dacă cheia dumneavoastră este protejată de o parolă, vă va fi solicitată introducerea acesteia. După aceasta, serverul Apache va porni, iar dumneavoastră vă veţi putea conecta securizat prin intermediul URL-ului `https://www.example.com`.

În cazul în care aţi compilat şi aţi instalat Apache chiar dumneavoastră, în multe dintre fişierele de configurare ale distribuţiei, veţi observa că directivele SSL sunt încadrate într-un bloc `<ifDefine SSL>`. Este permisă astfel pornirea condiţională a serverului în modul SSL. Dacă porniţi direct executabilul `httpd`, puteţi să-i asociaţi la pornire parametrul `-DSSL`. Puteţi să folosiţi de asemenea scriptul `apachectl` folosind comanda `apachectl startssl`. În fine, dacă doriţi să porniţi întotdeauna serverul Apache cu suport SSL, puteţi să îndepărtaţi pur şi simplu secţiunea `<ifDefine>` şi să porniţi Apache în maniera obişnuită.

Dacă nu reuşiţi să porniţi serverul, verificaţi jurnalul de erori Apache pentru indicaţii despre ce ar fi putut să funcţioneze greşit. De exemplu, dacă nu vă puteţi lega la port asiguraţi-vă că nu rulează deja şi o altă instanţă de server Apache. Trebuie, de asemenea, să aveţi drepturi de administrare pentru a vă putea lega la portul 443.

Rezumat

În acest capitol, v-au fost prezentate elementele fundamentale ale protocolului SSL şi ale modulului `mod_ssl`, un modul Apache care implementează suportul pentru SSL. V-a fost oferită o introducere în instalarea şi configurarea bibliotecii OpenSSL şi a modulului `mod_ssl`, precum şi asupra utilizării instrumentului din linie de comandă `openssl` pentru generarea şi administrarea certificatului şi a perechii de chei. Puteţi să accesaţi documentaţia referitoare la modulul `mod_ssl` la adresa `http://httpd.apache.org/docs-2.0/mod/mod_ssl.html`, urmând să găsiţi explicaţii amănunţite asupra sintaxei şi informaţii suplimentare despre configurare. Amintiţi-vă că protocolul SSL este doar o parte din administrarea unui server securizat, alături de toate actualizările de securitate, configurarea sistemului de operare, controlul accesului, securitatea fizică, şi aşa mai departe.

Întrebări şi răspunsuri

Î. *Pot folosi SSL în contextul hosturilor virtuale bazate pe nume?*

R. O întrebare care apare frecvent se referă la modalitatea de folosire a hostingului virtual bazat pe nume în contextul SSL. Răspunsul este că acest lucru nu este deocamdată posibil. Hosturile virtuale bazate pe nume depind de headerul `Host` al solicitării HTTP, în timp ce verificarea certificatului are loc în timpul stabilirii conexiunii SSL, moment în care nu poate fi trimisă nici o solicitare HTTP. Există un protocol pentru actualizarea unei conexiuni preexistente de tip HTTP la TLS, dar majoritatea browserelor din momentul de faţă nu l-au implementat (consultaţi articolul RFC 2817, aflat la adresa `http://www.rfc-editor.org/rfc/rfc2817.txt`).

Î. *Pot folosi protocolul SSL cu alte protocoale?*

R. Modulul `mod_ssl` implementează protocolul SSL sub forma unui filtru. Alte protocoale care folosesc acelaşi server Apache pot să utilizeze cu uşurinţă avantajele oferite de SSL.

Atelier

Atelierul este conceput să anticipeze posibile întrebări, să vă ajute să recapitulaţi cele învăţate şi să începeţi să vă puneţi cunoştinţele în practică.

Test

1. Numiţi trei cerinţe care trebuie îndeplinite pentru a comunica în siguranţă pe Internet.
2. Cum puteţi porni o instanţă de Apache cu supotul SSL activat?

Soluţii

1. Confidenţialitate, integritate, şi autentificare.
2. Folosiţi scriptul de control `apachectl` şi comanda `apachectl startssl`.

CAPITOLUL 28

Optimizarea şi reglarea serverului MySQL

Un server MySQL bine îngrijit este un server care rulează fericit şi fără incidente. Optimizarea sistemului constă în administrarea competentă a componentelor hardware, şi în reglarea aplicaţiilor software.

În acest capitol, veţi afla:

- sfaturi pentru optimizarea hardware şi software a serverului MySQL;
- parametri cheie de pornire a serverul MySQL;
- cum să utilizaţi comanda `OPTIMIZE TABLE`;
- cum să utilizaţi comanda `EXPLAIN`;
- cum să utilizaţi comanda `FLUSH` pentru a curăţa tabele, fişiere jurnal şi fişiere temporare;
- cum să utilizaţi comanda `SHOW` pentru a obţine informaţii despre baze de date, tabele şi fişiere index;
- cum să utilizaţi comanda `SHOW` pentru informaţii despre starea sistemului.

Construirea unei platforme optimizate

Proiectarea unei scheme de baze de date bine structurate şi normalizate nu este decât jumătate din labirintul optimizării. Cealaltă jumătate constă din construirea şi reglarea fină a unui server care să ruleze această bază de date. Gândiţi-vă la cele patru componente principale ale unui server: procesorul, memoria, mediile de stocare şi sistemul de operare. Fiecare dintre acestea trebuie să funcţioneze impecabil, în caz contrar nici cel mai mare volum de proiectare sau de programare neputând să convingă baza dumneavoastră de date să meargă mai repede!

- **procesorul (CPU)** — cu cât acesta este mai rapid, cu atât mai repede va putea MySQL să proceseze datele dumneavoastră. Nu există un secret real: pur şi simplu un procesor cu o frecvenţă de ceas de 3,0 GHz va funcţiona semnificativ mai rapid decât un procesor de numai 1,0 GHz. În condiţiile în care viteza de ceas a procesoarelor creşte exponenţial, la preţuri rezonabile, nu ar trebui să fie dificil să obţineţi produse de o calitate bună pentru banii pe care sunteţi dispuşi să îi cheltuiţi.
- **memoria** — puneţi cât de mult RAM puteţi în calculator. Nu puteţi avea niciodată suficient, iar preţurile memoriilor RAM se vor menţine scăzute în continuare, cel puţin în viitorul apropiat. O cantitate mare de RAM poate compensa un procesor cu o frecvenţă de ceas mai mică.

- **Hard discul** — un hard disc corespunzător nu numai că va avea o capacitate de stocare suficient de mare, dar va fi şi suficient de rapid pentru a corespunde cerinţelor unui server de baze de date şi a traficului care se va desfăşura prin acesta. O măsură importantă a vitezei unui hard disc este timpul său de acces, sau cantitatea de timp necesară pentru a se roti şi a identifica informaţia solicitată. Timpul de acces se măsoară în milisecunde, iar un timp de acces mediu este în jur de 8 sau 9 milisecunde. Când achiziţionaţi un asemenea mediu de stocare, asiguraţi-vă că este suficient de încăpător pentru a conţine toate datele pe care le veţi stoca în cele din urmă în baza dumneavoastră de date, precum şi suficient de rapid pentru a le accesa cu viteză.

- **Sistemul de operare**—Dacă folosiţi un sistem de operare lipsit de respect faţă de resursele dumneavoastră, nu aveţi decât două opţiuni: cumpăraţi suficiente resurse (în speţă, RAM) astfel încât acest aspect să nu mai conteze, sau folosiţi un sistem de operare care să nu vă acapareze toate resursele doar pentru a afişa ferestruici şi culori frumoase. De asemenea, dacă sunteţi binecuvântat cu un calculator cu mai multe procesoare, asiguraţi-vă că sistemul dumneavoastră de operare poate să lucreze cu acest calculator, şi să-l folosească la adevărata sa valoare.

Dacă sistemul dumneavoastră cuprinde componentele adecvate, înseamnă că aţi parcurs deja câţiva paşi pe drumul optimizării serverului.

Utilizarea funcţiei `benchmark()`

Puteţi face un test rapid al vitezei serverului dumneavoastră folosind funcţia MySQL `benchmark()` pentru a vedea cât timp durează procesarea unei expresii date. Puteţi folosi o expresie cât se poate de simplă, ca de exemplu `10 + 10`, dar şi ceva mai extravagant, cum este cazul extragerii de date.

Indiferent de rezultatul expresiei, rezultatul funcţiei `benchmark()` va fi întotdeauna 0. Scopul funcţiei `benchmark()` nu este returnarea rezultatului expresiei, ci aflarea timpului necesar repetării expresiei respective de un anumit număr de ori. De exemplu, următoarea comandă execută expresia `10 + 10` de un milion de ori:

```
mysql> select benchmark(1000000,10+10);
+--------------------------+
| benchmark(1000000,10+10) |
+--------------------------+
|                        0 |
+--------------------------+
1 row in set (0.14 sec)
```

Această comandă va determina executarea unei extrageri de date, de asemenea de un milion de ori:

```
mysql> select benchmark(1000000, extract(year from now()));
```

```
+-------------------------------------------+
| benchmark(1000000, extract(year from now())) |
+-------------------------------------------+
|                                         0 |
+-------------------------------------------+
1 row in set (0.20 sec)
```

Numărul important este timpul, exprimat în secunde, care este timpul utilizat pentru executarea funcţiei; primul test a durat 0,14 secunde, în timp ce al doilea a necesitat 0,20 secunde. S-ar putea să doriţi să rulaţi o aceeaşi expresie `benchmark()` de mai multe ori în timpul unei zile (când serverul dumneavoastră este solicitat în mod diferit) pentru a vă da seama mai precis cum se comportă serverul respectiv.

Opţiuni de pornire ale serverului MySQL

MySQL AB oferă numeroase informaţii referitoare la reglarea parametrilor serverului, majoritatea dintre aceşti parametri nefiind însă folosiţi vreodată de către un utilizator obişnuit. Astfel, pentru a nu vă încărca până peste poate cu informaţii, această secţiune va aborda doar câteva dintre cele mai frecvent utilizate opţiuni de pornire care contribuie la reglarea precisă a unui server MySQL.

Când porniţi MySQL, este încărcat un fişier de configurare cu numele `my.cnf`. Acest fişier conţine informaţii care variază de la numărul portului la dimensiuniea zonei tampon, buffer, dar care pot fi suprascrise prin diferite opţiuni de pornire din linia de comandă. La momentul instalării, fişierul `my.cnf` este plasat în directorul `/etc`, dar aveţi posibilitatea să specificaţi şi un director alternativ.

În directorul `support-files` din directorul de instalare al serverului dumneavoastră MySQL veţi găsi patru fişiere de configurare, fiecare reglat pentru un interval specific de memorie disponibilă:

- `my-small.cnf` — pentru sistemele cu mai puţin de 64 MB de RAM, în care serverul de baze de date MySQL este folosit numai ocazional.
- `my-medium.cnf` — pentru sistemele cu mai puţin de 64 MB de RAM, în care MySQL este activitatea principală a sistemului respectiv, sau pentru sistemele cu cel mult 128 MB de RAM, în care MySQL împarte spaţiul de rulare cu alte procese. Aceasta este cea mai frecvent folosită configurare, în care MySQL este instalat pe acelaşi sistem cu un server Web şi primeşte o cantitate moderată de trafic.
- `my-large.cnf` — pentru un sistem cu 128 MB până la 512 MB de RAM, a cărui activitate principală este rularea serverului de baze de date MySQL.
- `my-huge.cnf` — pentru un sistem cu 1 GB până la 2 GB de RAM, a cărui activitate principală este rularea serverului de baze de date MySQL.

Pentru a utiliza oricare dintre aceste fişiere primare de configurare este suficient să copiaţi fişierul selectat de dumneavoastră la adresa `/etc/my.cnf` (de fapt, în locul în care este plasat fişierul `my.cnf` pe sistem) şi să modificaţi orice informaţii specifice sistemului dumneavoastră, cum ar fi porturile sau calea spre diferite fişiere.

Parametri cheie de pornire

Doi dintre parametrii de pornire vor afecta cel mai mult performanţele sistemului dumneavoastră: `key_buffer_size` şi `table_cache`. Aşa că, dacă nu veţi reuşi să configuraţi corespunzător decât doi parametri, asiguraţi-vă că despre aceştia este vorba!

Valoarea parametrului `key_buffer_size` este dimensiunea zonei tampon folosită de către fişiere index. Cu cât aceasta este mai mare, cu atât mai rapid se va termina executarea comenzii SQL, şi cu atât mai rapid va fi returnat un rezultat. Încercaţi să găsiţi soluţia de mijloc dintre reglarea fină şi supra-optimizare: s-ar putea ca parametrul dumneavoastră `key_buffer_size` să fie de 256 MB corespunzând unui sistem cu 512 MB de RAM, dar o valoare mai mare de 256 MB ar putea duce la degradarea performanţelor serverului.

O modalitate simplă de verificare a performanţelor efective ale zonei tampon este examinarea a patru variabile suplimentare: `key_read_requests`, `key_reads`, `key_write_requests`, şi `key_writes`. Puteţi afla valorile acestor variabile prin intermediul comenzii `SHOW STATUS`:

```
mysql> show status;
```

Rezultatul acestei comenzi va fi o listă lungă de variabile şi de valori, prezentate în ordine alfabetică. Identificaţi liniile care arată în modul următor (valorile dumneavoastră vor diferi):

```
| Key_read_requests          | 602843    |
| Key_reads                  | 151       |
| Key_write_requests         | 1773      |
| Key_writes                 | 805       |
```

Dacă împărţiţi valoarea lui `key_read` la valoarea lui `key_reads_requests`, rezultatul ar trebui să fie mai mic decât `0,01`. De asemenea, dacă împărţiţi valoarea lui `key_write` la valoarea lui `key_writes_requests`, rezultatul ar trebui să fie mai mic decât `1`. Folosind valorile anterioare, avem rezultatele `0.000250479809834401` şi respectiv `0.454032712915962`, care se încadrează foarte bine în gama parametrilor acceptabili. Pentru a încerca să micşoraţi şi mai mult valorile, puteţi creşte valoarea lui `key_buffer_size`, dar aceste numere arată foarte bine aşa cum sunt ele.

Celălalt parametru important este `table_cache`, numărul de tabele deschise în toate firele de execuţie. Valoarea implicită este 64, dar s-ar putea să fie nevoie să o modificaţi. Cu ajutorul comenzii `SHOW STATUS`, căutaţi valoarea variabilei cu numele `open_tables`. Dacă numărul este mare, ar trebui să creşteţi valoarea parametrului `table_cache`.

Fişierele de configurare oferite drept exemplu folosesc diferite combinaţii ale parametrilor `key_buffer_size` şi `table_cache`, pe care le puteţi studia ca o bază de pornire pentru orice modificări care s-ar dovedi necesare. Ori de câte ori modificaţi configuraţia, va trebui să reporniţi serverul pentru ca aceste modificări să intre în vigoare. În orice caz, aveţi grijă să testaţi schimbările de configuraţie într-un mediu de dezvoltare, înainte de a le implementa în mediul de producţie.

Optimizarea structurii tabelelor

O structură optimizată a tabelelor este diferită de o tabelă bine proiectată. Optimizarea structurii tabelelor se referă la recuperarea spaţiului nefolosit rămas în urma ştergerii de date şi, în principiu, la curăţarea tabelei după efectuarea oricăror modificări ale structurii acesteia. Comanda `OPTIMIZE TABLE SQL` se ocupă de acest aspect, folosind următoarea sintaxă:

```
OPTIMIZE TABLE nume_tabela[,nume_tabela]
```

De exemplu, dacă doriţi să optimizaţi tabela `grocery_inventory` din baza de date `testDB`, folosiţi comanda:

```
mysql> optimize table grocery_inventory;
+--------------------------+----------+----------+----------+
| Table                    | Op       | Msg_type | Msg_text |
+--------------------------+----------+----------+----------+
| testDB.grocery_inventory | optimize | status   | OK       |
+--------------------------+----------+----------+----------+
1 row in set (0.08 sec)
```

Nu vi se prezintă în mod explicit reparaţiile efectuate, dar textul din coloana `Msg_text` arată că tabela `grocery_inventory` a fost într-adevăr optimizată. Dacă rulaţi din nou comanda de optimizare, textul se va modifica, demonstrându-şi utilitatea:

```
mysql> optimize table grocery_inventory;
+--------------------------+----------+----------+-----------------------------+
| Table                    | Op       | Msg_type | Msg_text                    |
+--------------------------+----------+----------+-----------------------------+
| testDB.grocery_inventory| optimize | status   | Table is already up to date |
+--------------------------+----------+----------+-----------------------------+
1 row in set (0.03 sec)
```

Trebuie să vă avertizăm că tabela este blocată în timpul procesului de optimizare, aşa că dacă este vorba de o tabelă mare, optimizarea ar trebui să fie efectuată în timpul unor perioade de revizie, sau când traficul pe sistem este foarte redus.

Optimizarea interogărilor

Optimizarea interogărilor are multe de-a face cu utilizarea corespunzătoare a fişierelor index. Comanda `EXPLAIN` va examina o instrucţiune `SELECT` dată, pentru a verifica dacă este optimizată la maxim, apelând la fişiere index ori de câte ori este posibil. Această acţiune este cât se poate de utilă în special în contextul unor interogări complexe, care folosesc mai multe operaţii de tip `JOIN`. Sintaxa comenzii `EXPLAIN` este:

```
EXPLAIN SELECT instructiune
```

Rezultatul rulării unei comenzi `EXPLAIN` este o tabelă de informaţii care conţin următoarele coloane:

- `table` — numele tabelei.
- `type` — tipul de intersectare, dintre opţiunile posibile.
- `possible_keys` — această coloană indică fişierele index care ar putea fi folosite de MySQL pentru a identifica liniile din tabelă. În cazul în care rezultatul este `NULL`, nu există nici un fişier index care ar putea fi de ajutor în contextul acestei interogări. Ar trebui să studiaţi în acest caz structura tabelei dumneavoastră pentru a vedea dacă nu puteţi crea vreun index care să crească performanţele interogării respective.
- `key` — cheia folosită efectiv în interogare sau `NULL` (dacă nu a fost utilizat nici un index).
- `key_len` — lungimea cheii folosite, dacă este cazul.
- `ref` — orice coloane folosite în contextul cheii pentru obţinerea unui rezultat.
- `rows` — numărul de linii pe care MySQL trebuie să le examineze pentru a executa interogarea.
- `extra` — informaţii suplimentare referitoare la modul în care MySQL va executa interogarea. Aveţi la dispoziţie câteva opţiuni, cum ar fi `Using index` (a fost folosit un index) şi `Where` (a fost folosită o clauză `WHERE`).

Rezultatul următoarei comenzi `EXPLAIN` ne sugerează executarea unei interogări neoptimizate:

```
mysql> explain select * from grocery_inventory;
+-------------------+------+---------------+------+---------+------+------+-------+
| table             | type | possible_keys | key  | key_len | ref  | rows | Extra |
+-------------------+------+---------------+------+---------+------+------+-------+
| grocery_inventory | ALL  | NULL          | NULL |    NULL | NULL |    6 |       |
+-------------------+------+---------------+------+---------+------+------+-------+
1 row in set (0.00 sec)
```

Nu sunt totuşi prea multe posibilităţi de a optimiza o interogare de tipul „selectează totul", cu excepţia adăugării unei clauze `WHERE` în contextul cheii primare. Coloana `possible_keys` va conţine atunci textul `PRIMARY`, iar coloana `Extra` ne va arăta acest lucru, afişând `Where used` (s-a folosit clauza `WHERE`).

Când folosiţi comanda EXPLAIN pentru interogări ce conţin secvenţe JOIN, o modalitate rapidă de estimare a optimizării interogării este să vă uitaţi la valorile din coloana rows. În exemplul precedent s-au obţinut numerele 2 şi 1. Înmulţiţi aceste numere şi veţi obţine rezultatul 2. Acesta este numărul de linii pe care MySQL trebuie să le parcurgă pentru a vă oferi rezultatul interogării. Este ideal ca acest număr să fie cât mai mic cu putinţă şi nu prea are cum să fie mai mic decât 2!

Pentru mult mai multe informaţii despre comanda EXPLAIN vă sugerăm să vizitaţi pagina corespunzătoare a manalului MySQL, la adresa `http://www.mysql.com/doc/E/X/EXPLAIN.html`.

Utilizarea comenzii FLUSH

Utilizatorii cu privilegii de reîncărcare pentru o anumită bază de date pot folosi comanda FLUSH pentru a curăţa fişierele temporare interne folosite de MySQL. Practica frecventă este rezervarea pentru utilizatorul de nivel root a permisiunilor necesare pentru executarea de comenzi administrative, cum este cazul comenzii FLUSH.

Sintaxa comenzii FLUSH este:

```
FLUSH optiuni_flush
```

Opţiunile cele mai frecvent utilizate ale comenzii FLUSH sunt următoarele:

- PRIVILEGES
- TABLES
- HOSTS
- LOGS

Aţi folosit deja comanda FLUSH PRIVILEGES pe parcursul acestei cărţi, după ce aţi adăugat utilizatori noi. Această comandă nu face decât să reîncarce tabelele de permisiuni în baza dumneavoastră de date MySQL, permiţând modificărilor să intre în vigoare fără oprirea şi repornirea serverului MySQL. Când lansaţi o comandă FLUSH PRIVILEGES, răspunsul Query OK vă va asigura că procesul de curăţare a avut loc fără nici o problemă.

```
mysql> flush privileges;
Query OK, 0 rows affected (0.10 sec)
```

Comanda FLUSH TABLES va închide toate tabelele deschise sau folosite în momentul respectiv şi, în principiu, îi va oferi serverului dumneavoastră MySQL o pauză de o milisecundă înainte de a-l trimite înapoi la lucru. Când fişierele temporare sunt goale, MySQL poate să folosească mai bine memoria pe care o are la dispoziţie. Din nou, speraţi să obţineţi răspunsul Query OK:

```
mysql> flush tables;
Query OK, 0 rows affected (0.21 sec)
```

Comanda FLUSH HOSTS acţionează în mod explicit pe tabelele temporare de hosturi. Dacă nu reuşiţi să vă conectaţi la serverul dumneavoastră MySQL, de cele mai multe ori cauza este că s-a atins numărul maxim de conexiuni din partea unui anumit host, urmată de generarea de mesaje de eroare. Când MySQL observă un număr mare de erori în acest context, el presupune că ceva este în neregulă şi blochează pur şi simplu orice alte încercări ulterioare de conectare din partea hostului respectiv. Comanda FLUSH HOSTS va opri acest proces şi va deschide astfel premisele unei conectări încununate de succes:

```
mysql> flush hosts;
Query OK, 0 rows affected (0.00 sec)
```

Comanda FLUSH LOGS închide şi redeschide toate fişierele jurnal. Dacă fişierul dumneavoastră jurnal ajunge de dimensiuni pantagruelice şi doriţi să deschideţi un fişier nou, această comandă vă va ajuta, creând un fişier jurnal nou, lipsit de conţinut. Parcurgerea unui fişier jurnal care acoperă un an calendaristic în căutare de erori poate fi cel puţin neplăcută, aşa că este de preferat să folosiţi această comandă cel puţin o dată pe lună.

```
mysql> flush logs;
Query OK, 0 rows affected (0.04 sec)
```

Utilizarea comenzii SHOW

Comanda SHOW poate fi folosită în câteva moduri diferite, generând rezultate de natură să vă prezinte un număr mare de informaţii utile despre bazele dumneavoastră de date MySQL, utilizatori şi tabele. În funcţie de nivelul dumneavoastră de acces, unele dintre comenzile SHOW nu vă vor fi accesibile, sau vă vor oferi numai informaţii reduse. Utilizatorul de nivel root are posibilitatea de a folosi toate comenzile SHOW, obţinând cele mai complete rezultate.

Printre cele mai frecvente solicitări de tip SHOW se numără următoarele, despre care veţi afla în curând mai multe detalii:

```
SHOW GRANTS FOR utilizator
SHOW DATABASES [LIKE ceva]
SHOW [OPEN] TABLES [FROM nume_baza_de_date] [LIKE ceva]
SHOW CREATE TABLE nume_tabela
SHOW [FULL] COLUMNS FROM nume_tabela [FROM nume_baza_de_date] [LIKE ceva]
SHOW INDEX FROM nume_tabela [FROM nume_baza_de_date]
SHOW TABLE STATUS [FROM nume_baza_de_date] [LIKE ceva]
SHOW STATUS [LIKE ceva]
SHOW VARIABLES [LIKE ceva]
```

Comanda SHOW GRANTS va afişa privilegiile deţinute de un anumit utilizator de la un host dat. Aceasta este o modalitate facilă prin care puteţi verifica statutul curent al unui utilizator, în special dacă aţi primit solicitarea de a modifica privilegiile utilizatorului respectiv. Cu ajutorul comenzii SHOW GRANTS puteţi să

vă asiguraţi dacă el nu are deja privilegiile solicitate. De exemplu, acestea sunt privilegiile deţinute de utilizatorul `joeuser`:

```
mysql> show grants for joe@localhost;
+-----------------------------------------------------------+
| Grants for joeuser@localhost                              |
+-----------------------------------------------------------+
| GRANT USAGE ON *.* TO 'joeuser'@'localhost'
IDENTIFIED BY PASSWORD '34f3a6996d856efd'                   |
| GRANT ALL PRIVILEGES ON testDB.* TO 'joeuser'@'localhost' |
+-----------------------------------------------------------+
```

Dacă nu sunteţi utilizator de nivel `root` sau chiar `joeuser`, veţi primi un mesaj de eroare. Cu excepţia cazului în care sunteţi utilizatorul de nivel `root`, nu puteţi să vedeţi decât informaţiile referitoare la utilizatorul dumneavoastră. De exemplu, utilizatorului `joeuser` nu i se permite să acceseze informaţiile despre utilizatorul de nivel `root`:

```
mysql> show grants for root@localhost;
ERROR 1044: Access denied for user:'joeuser@localhost' to database 'mysql'
```

Pe parcursul acestui capitol trebuie să vă asiguraţi asupra nivelului dumneavoastră de acces. În cazul în care nu sunteţi utilizator de nivel `root`, unele dintre comenzile pe care vi le vom prezenta nu vă vor fi disponibile, sau veţi obţine numai informaţii limitate.

Obţinerea de informaţii despre baze de date şi tabele

Aţi folosit un număr deloc de neglijat de comenzi `SHOW` pe parcursul acestei cărţi, pentru a vedea lista de baze de date şi de tabele stocate pe serverul dumneavoastră MySQL. Pentru a vă reaminti, comanda `SHOW DATABASES` se ocupă exact cu asta—enumeră toate bazele de date care se află pe serverul MySQL:

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| testDB             |
| mysql              |
+--------------------+
2 rows in set (0.00 sec)
```

După ce aţi selectat baza de date cu care doriţi să lucraţi, puteţi folosi de asemenea comanda `SHOW` pentru a enumera tabelele din respectiva bază de date. În acest exemplu vom folosi baza `testDB` (lista dumneavoastră de tabele ar putea să fie diferită):

```
mysql> show tables;
+--------------------+
| Tables_in_testDB   |
+--------------------+
| grocery_inventory  |
| email              |
| master_name        |
| myTest             |
| testTable          |
+--------------------+
5 rows in set (Ø.Ø1 sec)
```

Dacă adăugaţi parametrul OPEN la comanda dumneavoastră SHOW TABLES veţi obţine o listă a tuturor tabelelor din sistemul de cache al tabelelor, arătând de câte ori au fost plasate tabelele în acest sistem şi de câte ori au fost folosite aceste tabele:

```
mysql> SHOW OPEN TABLES;
+-------------------------+--------------------+
| Open_tables_in_testDB   | Comment            |
+-------------------------+--------------------+
| grocery_inventory       | cached=1, in_use=Ø |
| email                   | cached=1, in_use=Ø |
| testTable               | cached=1, in_use=Ø |
| master_name             | cached=1, in_use=Ø |
| myTest                  | cached=1, in_use=Ø |
+-------------------------+--------------------+
5 rows in set (Ø.ØØ sec)
```

Folosind aceste informaţii în conjuncţie cu comanda FLUSH TABLES pe care aţi învăţat-o mai devreme în acest capitol, veţi putea să menţineţi operabilitatea bazei dumneavoastră de date. Dacă rularea comenzii SHOW OPEN TABLES vă arată cu tabelele sunt introduse de multe ori în sistemul de cache, dar nu sunt folosite în momentul respectiv, este recomandabil să folosiţi comanda FLUSH TABLES pentru a elibera memoria consumată de ele.

Obţinerea de informaţii despre structura tabelelor

O comandă foarte utilă este SHOW CREATE TABLE, care face exact ceea ce îi spune numele — vă arată instrucţiunea SQL utilizată pentru crearea unei anumite tabele:

```
mysql> show create table grocery_inventory;
+--------------------+-------------------------------------------+
| Table              | Create Table
|+-------------------+-------------------------------------------+
| grocery_inventory  | CREATE TABLE 'grocery_inventory' (
                     'id' int(11) NOT NULL auto_increment,
                     'item_name' varchar(5Ø) NOT NULL default '',
                     'item_desc' text,
                     'item_price' float NOT NULL default  Ø',
                     'curr_qty' int(11) NOT NULL default 'Ø',
                     PRIMARY KEY  ('id')
                     ) TYPE=MyISAM
+--------------------+-------------------------------------------+
1 row in set (Ø.ØØ sec)
```

În principiu, informaţiile sunt similare celor pe care le-aţi obţine dacă aţi studia schema tabelei, dar comanda SHOW CREATE TABLE are avantajul că poate fi utilizată cu rapiditate în cazul în care căutaţi doar să vă amintiţi cum se procedează, sau în care aveţi nelămuriri asupra modului în care a fost creată o anumită tabelă.

Dacă trebuie să cunoaşteţi structura tabelei, dar nu aveţi nevoie neapărat de comanda SQL folosită pentru crearea sa, puteţi utiliza comanda SHOW COLUMNS:

```
mysql> show columns from grocery_inventory;
+------------+-------------+------+-----+---------+----------------+
| Field      | Type        | Null | Key | Default | Extra          |
+------------+-------------+------+-----+---------+----------------+
| id         | int(11)     |      | PRI | NULL    | auto_increment |
| item_name  | varchar(50) |      |     |         |                |
| item_desc  | text        | YES  |     | NULL    |                |
| item_price | float       |      |     | 0       |                |
| curr_qty   | int(11)     |      |     | 0       |                |
+------------+-------------+------+-----+---------+----------------+
5 rows in set (0.00 sec)
```

Apropo

Comenzile SHOW COLUMNS şi DESCRIBE sunt aliasuri ele între ele, şi, de aceea, fac exact acelaşi lucru.

Comanda SHOW INDEX va afişa informaţii despre toate indexurile prezente într-o anumită tabelă. Sintaxa sa este:

```
SHOW INDEX FROM nume_tabela [FROM nume_baza_de_date]
```

Rezultatul rulării comenzii este un tabel plin cu informaţii, pornind de la numele coloanei până la cardinalitatea indexului. Coloanele returnate de acestă comandă sunt descrise în tabelul 28.1.

TABELUL 28.1. Coloanele obţinute prin comanda SHOW INDEX.

Numele coloanei	Descriere
Table	Numele tabelei.
Non_unique	1 sau 0.
	1 = indexul poate conţine duplicate.
	0 = indexul nu poate conţine duplicate.
Key_name	Numele indexului.
Seq_in_index	Numărul secvenţei de coloană al indexului; începe la 1.
Column_name	Numele coloanei.

TABELUL 28.1. (*continuare*)

Numele coloanei	Descriere
`Collation`	Ordinea de sortare a coloanelor; poate fi `A` (ascendent) sau `NULL` (nesortate).
`Cardinality`	Numărul de valori unice din index.
`Sub_part`	Într-o coloană parțial indexată, aceasta reprezintă numărul de caractere indexate, sau NULL dacă este indexată întreaga cheie.
`Packed`	Dimensiunea coloanei numere.
`Comment`	Orice alte comentarii.

O altă comandă care produce o ca rezultat un tabel cu foarte multe informații este comanda `SHOW TABLE STATUS`. Sintaxa acestei comenzi este:

```
SHOW TABLE STATUS [FROM nume_baza_de_date] LIKE 'ceva'
```

Rezultatul rulării comenzii este un tabel plin cu informații, pornind de la dimensiunea și numărul de linii din tabelă până la următoarea valoare care va fi folosită într-un câmp de tipul `auto_increment`. Coloanele întoarse de acestă comandă sunt descrise în tabelul 28.2.

TABELUL 28.2. Coloanele obținute prin comanda `SHOW TABLE STATUS`.

Numele coloanei	Descriere
`Name`	Numele tabelei.
`Type`	Tipul de tabelă: MyISAM, BDB, InnoDB sau Gemini.
`Row_format`	Formatul de stocare: fix, dinamic sau comprimat.
`Rows`	Numărul de rânduri.
`Avg_row_length`	Lungimea medie a rândurilor.
`Data_length`	Lungimea fișierului de date.
`Max_data_length`	Lungimea maximă a fișierului de date.
`Index_length`	Lungimea fișierului index.
`Data_free`	Numărul de biți alocați, dar nefolosiți
`Auto_increment`	Următoarea valoare care va fi folosită într-un câmp `auto_increment`.
`Create_time`	Data și ora la care a fost creată tabela (în format `datetime`).
`Update_time`	Data și ora la care a fost actualizat ultima oară fișierul de date (în format `datetime`).

Tabelul 28.2. (*continuare*)

Numele coloanei	Descriere
`Check_time`	Data şi ora la care a fost verificată ultima oară tabela (în format `datetime`).
`Create_options`	Orice alte opţiuni folosite în instrucţiunea `CREATE TABLE`.
`Comment`	Orice alte comentarii adăugate la crearea tabelei. În plus, tabelele InnoDB folosesc această coloană pentru a raporta spaţiul disponibil în `tablespace`.

Obţinerea statutului sistemului

Comenzile `SHOW STATUS` şi `SHOW VARIABLES` vă vor oferi cu promptitudine informaţii importante despre serverul dumneavoastră de baze de date. Sintaxa acestor comenzi este pur şi simplu `SHOW STATUS` sau `SHOW VARIABLES` — nimic ieşit din comun.

Rezultatul rulării comenzii `SHOW STATUS` este o enumerare a nu mai puţin de 54 de variabile de stare, dar, dintre acestea, cele mai utile sunt:

- `Aborted_connects` — numărul de încercări nereuşite de conectare la serverul MySQL. Ori de câte ori vă confruntaţi cu o conectare nereuşită, ar trebui să investigaţi problema. Poate fi vorba fie despre un nume de utilizator sau o parolă eronată dintr-un script, sau poate numărul permis de conexiuni simultane este configurat la o valoare prea mică.
- `Connections` — numărul agregat de încercări de conectare la serverul MySQL de la ultima repornire a acestuia.
- `Max_used_connections` — numărul maxim de conexiuni care au fost utilizate simultan de la ultima repornire a serverului MySQL.
- `Slow_queries` — numărul de interogări care au durat mai mult decât `long_query_time`, variabilă a cărei valoare implicită este de 10 secunde. Dacă aveţi mai multe asemenea interogări lente, este timpul să vă investigaţi sintaxa SQL!
- `Uptime` — numărul total de secunde de rulare a serverului de la ultima sa repornire.

O listă completă a variabilelor `SHOW STATUS`, împreună cu explicaţii detaliate despre valorile acestora sunt disponibile în Manualul de MySQL, aflat la adresa Web `http://www.mysql.com/doc/S/H/SHOW_STATUS.html`.

Comanda `SHOW VARIABLES` produce mai multe rezultate chiar şi decât `SHOW STATUS` — aproximativ 82! Variabilele oferite prin rularea comenzii `SHOW VARIABLES` controlează operarea generală a serverului MySQL şi printre ele se numără următoarele:

- `connect_timeout` — numărul de secunde pe care serverul MySQL le va aştepta în contextul încercării de conectare înainte de a renunţa;
- `have_innodb` — rezultatul va fi `YES` dacă MySQL permite lucrul cu tabele `InnoDB`;
- `have_bdb` — rezultatul va fi `YES` dacă MySQL permite lucrul cu tabele Berkeley DB;
- `max_connections` — numărul maxim de conexiuni simultane posibile la serverul MySQL înainte ca acesta să refuze noi conexiuni;
- `port` — portul pe care rulează MySQL;
- `table_type` — tipul implicit de tabelă pentru MySQL, de obicei `MyISAM`;
- `version` — numărul versiunii de MySQL.

Puteţi obţine o listă completă a variabilelor `SHOW VARIABLES`, împreună cu explicaţii detaliate despre valorile acestora din Manualul de MySQL, aflat la adresa Web `http://www.mysql.com/doc/S/H/SHOW_VARIABLES.html`. După ce aflaţi valorile corespunzătoare serverului dumneavoastră, puteţi să le modificaţi în fişierul de configurare al MySQL sau în comanda de lansare a serverului.

Rezumat

Calea către un server MySQL optimizat începe cu componentele hardware şi cu sistemul de operare care rulează pe acestea. Procesorul sistemului dumneavoastră ar trebui să aibă o frecvenţă de ceas suficient de rapidă şi ar trebui să fie secondat de o cantitate de memorie RAM suficient de mare pentru a-l ajuta în momentele grele. Acest lucru este cu atât mai important în condiţiile în care serverul MySQL trebuie să împartă aceste resurse cu alte procese, cum ar fi, de exemplu, un server Web. În plus, hard discul este şi el important, deoarece o capacitate de stocare mică va limita cantitatea de informaţii pe care le puteţi plasa în baza dumneavoastră de date. Timpul de acces al hard discului este de asemenea important - un timp de acces crescut va duce la scăderea globală a performanţelor serverului. Este preferabil ca sistemul de operare pe care îl folosiţi să nu încarce calculatorul şi să împartă resursele cu serverul MySQL în loc să le folosească în mod masiv.

Câţiva dintre parametrii cheie de lansare a serverului MySQL sunt `key_buffer_size` şi `table_cache`, dar nu sunt singurii. Valorile comun utilizate sunt disponibile în exemplele de fişiere de configurare ale MySQL, sau puteţi pur şi simplu să modificaţi valorile acestor variabile şi să urmăriţi performanţele serverului pentru a vederea dacă obţineţi rezultatele pe care vi le doriţi pentru mediul dumneavoastră de lucru.

După optimizarea componentelor hardware şi a aplicaţiilor software urmează optimizarea tabelelor, precum şi a interogărilor de tip SELECT. Optimizarea tabelelor, cu ajutorul comenzii OPTIMIZE, vă permite să câştigaţi spaţiul de memorie ocupat, dar nefolosit. Puteţi vedea cât de bine sunt optimizate (sau nu) interogările dumneavostră folosind comanda EXPLAIN. Rezultatul rulării acesteia vă va arăta dacă şi cum folosiţii indexurile, precum şi dacă puteţi utiliza anumiţi indecşi pentru a creşte viteza interogării respective.

Atenţia pe care o acordaţi serverului dumneavostră MySQL vă va asigura că acesta continuă să ruleze fără incidente. Comenzi administrative precum FLUSH şi SHOW vă vor ajuta să reunoaşteţi şi să reparaţi rapid problemele potenţiale. Toate aceste comenzi sunt proiectate pentru a-i da serverului MySQL o milisecundă de pauză dacă acesta este foarte încărcat. Un număr mare de comenzi de tip SHOW vă stă la dispoziţie, afişându-vă informaţii despre structura bazelor de date, a tabelelor, a indexurilor, precum şi a nivelului de performanţă a sistemului dumneavoastră.

Întrebări şi răspunsuri

Î. Poate să folosească MySQL sistemele cu mai multe procesoare?

R. Sigur că da. Dacă sistemul dumneavoastră de operare ştie să lucreze cu mai multe procesoare, MySQL le va folosi şi el, la rândul său. Performanţele şi nivelul de optimizare al serverului MySQL rulând pe un sistem multi-procesor variază în funcţie de sistemul de operare. Pentru a afla mai multe informaţii pertinente la sistemul dumneavoastră de operare, consultaţi secţiunea corespunzătoare din manualul de MySQL, aflată la adresa: `http://www.mysql.com/doc/O/p/Operating_System_Specific_Notes.html`

Î. Ce nivel de acces trebuie să am pentru a putea folosi comanda `OPTIMIZE`?

R. Orice utilizator care are privilegii `INSERT` asupra unei tabele poate rula comenzi `OPTIMIZE`. Dacă un utilizator are numai permisiunea `SELECT`, comanda `OPTIMIZE` nu va fi executată.

Atelier

Atelierul este conceput să anticipeze posibile întrebări, să vă ajute să recapitulaţi cele învăţate şi să începeţi să vă puneţi cunoştinţele în practică.

Test

1. Ce funcţie MySQL vă permite să rulaţi o expresie de mai multe ori, perntru a afla viteza iteraţiilor?
2. Ce comandă SQL va curăţa structura tabelelor dumneavoastră?
3. Ce comandă `FLUSH` va reseta fişierele jurnal MySQL?
4. Pentru a identifica rapid dacă MySQL poate lucru cu tabelele de tip `InnoDB` aţi folosi comanda `SHOW STATUS` sau comanda `SHOW VARIABLES`?
5. Scrieţi o instrucţiune SQL care să vă permite să vedeţi instrucţiunile SQL utilizate la crearea unei tabele numită `myTable`.

Soluţii

1. Funcţia `benchmark()`.
2. `OPTIMIZE`.
3. `FLUSH LOGS`.

4. `SHOW VARIABLES`.
5. `SHOW CREATE TABLE myTable`.

Activităţi

1. Dacă aveţi acces de nivel `root` la server, modificaţi valorile parametrilor `key_buffer_size` şi `table_cache` şi rulaţi funcţia `benchmark()` după fiecare modificare pentru a vedea cum diferă timpii de execuţie.
2. Folosiţi comanda `OPTIMIZE` asupra tuturor tabelelor pe care le-aţi creat în baza dumneavoastră de date, pentru a rezolva orice probleme de structură a acestora.
3. Folosiţi comanda `SHOW STATUS` pentru a obţine informaţii despre serverul dumneavoastră MySQL, după care comanda `FLUSH` pentru a elibera serverul. După fiecare comandă, folosiţi din nou `SHOW STATUS` pentru a vedea ce comenzi afectează anumite rezultate afişate prin rularea `SHOW STATUS`.

CAPITOLUL 29
Actualizări de software

Pe tot parcursul acestei cărţi vi s-a reamintit să căutaţi informaţii referitoare la versiuni noi de PHP, Apache şi MySQL şi să fiţi atenţi la posibila necesitate a actualizării acestora. De asemenea, vi s-a explicat cum puteţi să adăugaţi diferite funcţii noi limbajului PHP la momentul compilării acestuia, dar numai în contextul instalării aplicaţiei software.

În acest scurt capitol, veţi învăţa cum să actualizaţi aplicaţia dumneavoastră deja instalată, fără a crea un haos total în sistem, inclusiv cum să:

- fiţi la curent cu noile versiuni de software;
- actualizaţi serverul de baze de date MySQL;
- actualizaţi serverul Web Apache;
- actualizaţi limbajul PHP.

Cum să rămâneţi în formă

Până acum ar fi trebui să aveţi deja trecute în lista dumneavoastră de „semne de carte" virtuale site-urile Web corespunzătoare aplicaţiilor Apache, PHP şi MySQL. Indiferent dacă folosiţi aceste tehnologii numai de o zi, sau dacă le-aţi folosit pe parcursul ultimilor şase ani — întotdeauna se va ivi un motiv să vă întoarceţi la site-uri. Motivul principal de vizitare a acestor site-uri Web este necesitatea obţinerii de actualizări pentru aplicaţiile deja instalate pe sitemul dumneavoastră. Desigur, puteţi oricând să vă înscrieţi la o listă de mesaje e-mail care distribuie numai acest tip de ştiri:

- pentru ştiri referitoare la MySQL, mergeţi la adresa `http://lists.mysql.com/` şi înscrieţi-vă la lista de mesaje MySQL Announcements (anunţuri MySQL).
- pentru ştiri referitoare la Apache, mergeţi la adresa `http://www.apache.org/foundation/mailinglists.html` şi înscrieţi-vă la lista de mesaje Apache News and Announcements (ştiri şi anunţuri Apache).
- pentru ştiri referitoare la PHP, mergeţi la adresa `http://www.php.net/mailing-lists.php` şi înscrieţi-vă la lista de mesaje Announcements (anunţuri).

Când să actualizați

Așa cum vi s-a explicat în capitolele referitoare la instalare, versiuni minore sunt lansate ori de câte ori dezvoltatorii simt nevoia să facă aceasta — fără nici un program special. Doar pentru că s-a lansat însă o nouă versiune minoră nu înseamnă în mod explicit că ar trebui să vă grăbiți să vă actualizați aplicația. Alteori, asta este exact ceea ce ar trebui să faceți.

Cel mai important moment în care ar trebui să vă actualizați imediat aplicația dumnevoastră software este o dată cu anunțarea rezolvării unei probleme de securitate. De obicei, problemele de securitate nu sunt descoperite decât în momentul în care ele încep să fie exploatate — uneori în mediul de testare, dar alteori de către un utilizator lipsit de scrupule, care dorește să cauzeze probleme. Puteți fi sigur că apariția unei probleme de securitate este o prioritate de cel mai înalt nivel pentru dezvoltatori și că veți fi anunțat în cel mai scurt timp despre apariția unei actualizări care să rezolve respectiva problemă. În acest moment ar trebui să începeți imediat procedura de instalare a pachetului respectiv — chiar dacă nu folosiți în mod special elementul cauzator de probleme. O gaură de securitate este totuși o gaură — de ce să o lăsați neacoperită?

Iată un exemplu dintr-un jurnal de modificări aduse codului Apache, care documentează o modificare apărută între versiunile 2.0.46 și 2.0.47, exemplu de indicator al necesității de actualizare:

```
SECURITY [CAN-2003-0254]: Fixed a bug in ftp proxy causing denial of service when
target host is IPv6 but proxy server can't create IPv6 socket. Fixed by the reporter.

SECURITY [CAN-2003-0254]: A fost rezolvată o problemă în proxy-ul de fpt care permite
un atac de tipul denial of service când hostul țintă este IPv6, dar serverul proxy nu
poate crea un socket IPv6. Problemă rezolvată de cel care a raportat-o.
```

O regulă simplă ar fi că, dacă în jurnalul de modificări apare, în orice formă, cuvântul *securitate*, ar trebui să vă actualizați aplicația.

Dacă noua lansare este pur și simplu o versiune de întreținere, ceea ce înseamnă că rezolvă probleme și oferă îmbunătățiri generale de diverse procese care au loc în timpul dezvoltării normale, foarte probabil că nu trebuie să lăsați totul baltă și să începeți procesul de actualizare. Iată câteva exemple de elemente de întreținere, din jurnalele de modificări ale Apache și PHP.

```
Add support for "streamy" PROPFIND responses. (from Apache changelog)
Fixed bug 21958 (workaround for unusual realpath() on AIX & Tru64). (from PHP changelog)

Implementare de răspunsuri PROPFIND de tip "streamy" (din jurnalul de modificări Apache)
Rezolvat eroare 21958 (în contextul funcționării neobișnuite a realpath() pe AIX & Tru64).
(din jurnalul de modificări PHP)
```

Dacă această listă de modificări nu conține elemente relevante pentru dumneavoastră, activitatea sau mediul dumneavoastră de lucru, probabil că puteți amâna actualizarea până la următoarea dată la care sunt programate activitățile de întreținere, sau până la următoarea zi ploioasă. De exemplu, dacă toate erorile rezolvate într-o nouă versiune de întreținere a PHP se referă la platforma AIX sau Tru64, iar dumneavoastră rulați aplicația pe o arhitectură Linux sau Intel, nu aveți nici un motiv să o actualizați.

Chiar dacă nu actualizaţi imediat aplicaţia, o idee bună este să rămâneţi cel mult cu o versiune sau două în urma versiunii curente de producţie. Este foarte posibil ca orice versiune anterioară să rateze facilităţi noi, sau rezolvări de probleme care sunt într-adevăr relevante în contextul activităţii sau mediul dumneavoastră de lucru.

Actualizarea serverului MySQL

Indiferent dacă pe sistemul dumneavoastră rulaţi Linux/Unix sau Windows, actualizarea versiunilor minore de MySQL este foarte facilă — instalaţi noua versiune ca şi cum aplicaţia deja instalată nici nu ar exista.

Apropo

Înainte de a actualiza serverul MySQL la o versiune minoră nouă ar trebui să creați o copie back-up a bazelor de date care rulează pe server.

MySQL AB a făcut foarte uşoară actualizarea în cadrul aceleiaşi versiuni de bază; 4.0 este o asemenea versiune, 4.1 va fi şi ea o versiune de bază şi aşa mai departe. Astfel, actualizarea la orice versiune minoră din familia 4.0.x este la fel de facilă ca instalarea unei aplicaţii noi peste vechiul software. Programul de instalare de sub mediul Windows face tot acest proces invizibil, dar exact asta are loc. Utilizatorii de Linux/Unix care instalează din distribuţia binară nu trebuie decât să reconecteze directorul simbolic `"mysql"` la noua distribuţie, dezarhivată, ca parte a procesului de instalare.

Dacă procesul de actualizare vă pune probleme, puteţi citi sfaturile şi rezolvările oferite la adresa `http://www.mysql.com/doc/en/Upgrade.html`. Totuşi, actualizarea la versiuni minore de MySQL a fost întotdeauna o experienţă deloc neplăcută pentru mine, indiferent platformă.

Actualizarea serverului Apache

Similar serverului de baze de date MySQL, actualizarea sau recompilarea serverului Web Apache urmează acelaşi proces ca instalarea iniţială a aplicaţiei respective. Utilizatorii de Windows au avantajul existenţei unui program de instalare, care va detecta automat existenţa versiunii anterioare, o va îndepărta şi va instala noua versiune.

Pentru utilizatorii care rulează un sistem de operare de tipul Linux/Unix, procesul urmează de asemenea aceleaşi direcţii ca instalarea originală. Când dezarhivaţi noua distribuţie, aceasta va crea un director care va purta numele noii versiuni. De exemplu, dacă versiunea dumneavoastră anterioară a fost 2.0.46 şi actualizaţi la versiunea 2.0.47, directoarele dumneavoastră se vor numi `httpd-2.0.46` şi, respectiv, `httpd-2.0.47`.

Directorul propriu-zis de instalare pentru Apache este ales de dumneavoastră, când rulaţi scriptul `configure`; de exemplu,

```
# ./configure --prefix=/usr/local/apache2
```

După ce rulaţi scriptul `configure` pentru a compila noua versiune de Apache este suficient să parcurgeţi procesele `make` şi `make install`, exact aşa cum aţi procedat când aţi instalat prima oară serverul Apache.

Puteţi inclusiv să instalaţi noua versiune de Apache direct peste versiunea anterioară — chiar şi cu binarul `http` rulând. Trebuie numai să vă asiguraţi că v-aţi copiat fişierele de configurare, în cazul în care lucrurile nu vor funcţiona chiar aşa cum trebuie. Dacă vă simţiţi însă mai bine instalând versiunea nouă într-un director separat, din nou nu este nici o problemă — trebuie numai să vă mutaţi toate fişierele corespunzătoare (toate fişierele din directorul `document`) în noul director şi să editaţi în mod corespunzător noul fişier `httpd.conf`. Metoda cea mai bună este cea pe care o preferaţi dumneavoastră; pur şi simplu una dintre ele necesită mai multe mişcări de fişiere şi reconfigurări decât cealaltă.

După ce aţi actualizat serverul Apache pe un sistem Linux/Unix, ar trebui de asemenea să recompilaţi modulul PHP. Utilizatorii care rulează Windows nu au ce modul să recompileze, dar ar trebui să se asigure că modificările impuse de PHP sunt încă prezente în fişierul `httpd.conf`, cu referire directă la încărcarea modulului din structura de directoare a limbajului PHP.

Modificarea serverului Apache fără a-l actualiza

Să presupunem că doriţi să adăugaţi sau să îndepărtaţi diferite funcţii din Apache, dar că nu îl actualizaţi la o versiune minoră mai nouă decât cea instalată. Un exemplu ar fi adăugarea unui modul nou, sau actualizarea versiunii de bibliotecă de funcţii OpenSSL folosită de sistemul dumneavoastră.

În acest caz, utilizatorii de Linux/Unix ar trebui să ajungă la directorul sursă (de exemplu, `httpd-2.0.47`) şi să tasteze în lina de comandă `make clean`. Rezultatul acestei comenzi va fi, în esenţă, aducerea fişierelor de compilare la valorile lor iniţiale pentru ca dumneavoastră să puteţi recompila serverul Apache fără să vă mai împiedicaţi de valorile folosite anterior şi păstrate de acesta. După executarea comenzii `make clean` rulaţi scriptul `configure` cu noii parametri şi repetaţi procesele `make` şi `make install`. În această situaţie, nu trebuie să recompilaţi modulul PHP.

Actualizarea limbajului PHP

Întrucât utilizatorii de Linux/Unix pot să adauge atât de multe funcţii suplimentare aplicaţiei PHP parcurgând pur şi simplu diferitele opţiuni de compilare, este foarte probabil că vor actualiza sau vor modifica această aplicaţie mai des decât o vor face pentru Apache sau MySQL. Indiferent dacă actualizaţi la o nouă versiune minoră sau adăugaţi pur şi simplu funcţii suplimentare (sau le dezinstalaţi pe cele de care nu mai aveţi nevoie), procesul de modificare sau de

actualizare este identic cu procesul pe care l-aţi parcurs când aţi instalat pentru prima oară aplicaţia: `configure`, `make`, `make install`. Pasul `make install` va plasa modulele PHP în poziţia corespunzătoare din structura de directoare a instalării de Apache. O dată ce noul dumneavoastră modul este la locul său, reporniţi serverul Apache — noua versiune de PHP ar trebui în acest moment să poată fi utilizată.

Utilizatorii de Windows trebuie să parcurgă un proces diferit; adăugarea de funcţii noi la un modul preexistent nu solicită decât activarea modulului prin modificarea fişierului `php.ini` şi repornirea serverului. Actualizarea la o versiune minoră vă solicită descărcarea unui nou fişier de distribuţie. Conţinutul acestui fişier este apoi extras într-un director cu numele versiunii pe care o conţine. Trebuie, în continuare, să urmaţi paşii instalării în ceea ce priveşte localizarea corectă a fişierelor `php.ini` şi `php4ts.dll`, deoarece fiecare versiune produce un fişier diferit. În continuare, modificaţi orice cale de directoare din fişierul `httpd.conf` al serverului Apache pentru a corespunde noii instalări de PHP şi reporniţi serverul—noua versiune de PHP ar trebui în acest moment să poată fi utilizată.

Rezumat

Acest scurt capitol v-a oferit câteva linii directoare pentru menţinerea la zi a aplicaţiilor instalate pe sistemul dumneavoastră: MySQL, Apache şi PHP. Aţi învăţat unde să căutaţi noile actualizări şi cum să cântăriţi importanţa actualizării la o versiune mai nouă. La final, aţi învăţat pas cu pas procesele de actualizare sau de modificare a aplicaţiilor MySQL, Apache şi PHP.

Atelier

Atelierul este conceput să anticipeze posibile întrebări, să vă ajute să recapitulaţi cele învăţate şi să începeţi să vă puneţi cunoştinţele în practică.

Test

1. Care este considerat motivul principal al actualizării unei aplicaţii software la o versiune minoră?
2. Ce comandă va elimina fişierele `makefiles` anterioare, precum şi configurările păstrate în sistemul cache?

Soluţii

1. Probleme de securitate care au fost identificate şi rezolvate de către dezvoltatori.
2. `make clean`.

PARTEA a VII-a

Gânduri spre viitor

Capitolul 30. Facilitățile noi şi compatibilitatea cu versiunile anterioare ale PHP 5.0 541

Capitolul 31. Facilitățile noi şi compatibilitatea cu versiunile anterioare ale MySQL 4.1 547

CAPITOLUL 30
Facilitățile noi și compatibilitatea cu versiunile anterioare ale PHP 5.0

Toate elementele de PHP pe care vi le-am prezentat în această carte se bazează pe familia de versiuni 4, care a fost în producție de mai mulți ani. În acest timp, dezvoltatorii de la PHP Group și de la Zend Technologies au lucrat la următoarea generație PHP, versiunea 5. Având în vedere cât de complex și plin de facilități a fost limbajul PHP încă de la scrierea sa, este greu să ne imaginăm ce ar putea conține următoarea generație.

În acest scurt capitol, veți afla câteva informații despre facilitățile nou introduse în PHP 5, incluzând:

- o trecere în revistă a noului model de obiecte;
- o trecere în revistă a altor facilități nou introduse;
- când să luați în considerare trecerea la PHP 5.

Care este problema cu PHP 4?

„Problemele" familiei de versiuni 4 a limbajului PHP depind de persoana care le enumeră. Personal, eu nu văd nici o problemă cu modul în care lucrează PHP 4 deoarece:

- este rapid;
- este suficient de flexibil pentru a face ceea ce doresc să facă, continuând să rămână rapid.

Recunosc că acestea nu sunt dintre cele mai profesionale motive, dar ele prezintă până la urmă un punct important: dacă o tehnologie corespunde necesităților dumneavoastră, cum poate ea să fie „greșită"? Răspunsul este că nu, nu poate — cel puțin din punctul meu de vedere. Dar dacă ați ajuns la PHP din alte limbaje de programare, trecerea la un limbaj procedural flexibil, care se întâmplă să implementeze și principiile de programare orientată pe obiecte poate fi frustrantă. Desigur, ușurința de utilizare și robustețea sa, deși uneori ascunsă, sunt motive pentru care programatorii noi sunt atrași de la început de PHP — curba de învățare nu este deloc abruptă, iar sarcinile pot fi rezolvate.

Limbajul PHP are însă în acest context o problemă de marketing şi de poziţionare în contextul IT actual. Există voci care nu consideră că limbajul PHP ar fi potrivit pentru dezvoltarea de aplicaţii la nivel de companie, deoarece nu este un limbaj de programare testat în timp, structurat, orientat pe obiecte, cum este cazul limbajului C sau chiar Java. Poate că nu este cel mai bun moment ca un dezvoltator să încerce să-şi convingă managerii altfel decât prezentându-le exemple de cod PHP executând aceleaşi sarcini ca un cod C sau Java, cu acelaşi nivel de structurare, securitate, capacitate de refolosire şi interceptare de excepţii. Din această stare de fapt şi din alte probleme a apărut noua cale de dezvoltare a PHP 5, limbaj de programare al cărui scop principal este îmbunătăţirea modelului de obiecte, impunerea unei discipiline a scrierii de cod şi proiectarea specifică a unei versiuni de PHP care să corespundă cerinţelor dezvoltatorilor de cod orientat pe obiecte şi să le permită interfaţarea cu Java, .NET şi alte cadre de proiectare de aplicaţii la nivel de companie.

Ce se va întâmpla însă cu utilizatorii obişnuiţi, care nu doresc să aibă nimic de-a face cu obiecte, clase, .NET sau XML — doresc numai un limbaj de programare simplu, rapid, flexibil, care să-i ajute să creeze site-uri Web simple şi dinamice? Aceşti dezvoltatori nu sunt nici pe departe uitaţi. Deşi PHP 5 conţine un motor intern de scriptare foarte puternic şi un cadru de proiectare orientat pe obiecte mult îmbunătăţit, Grupul PHP şi Zend Technologies recunosc şi preţuiesc rădăcinile limbajului PHP şi grupul central de utilizatori care i-au asigurat omniprezenţa. Ca şi PHP 4, nici PHP 5 nu vă obligă să folosiţi acele elemente de limbaj pe care nu doriţi să le utilizaţi. Dacă aplicaţiile dumneavoastră sunt scrise la nivel scăzut, procedural, aşa cum v-a fost prezentat în această carte şi dacă acest nivel corespunde cerinţelor dumneavoastră, nu trebuie să modificaţi nimic. Dacă aveţi cunoştinţe de programare orientată pe obiecte, ar trebui să citiţi în secţiunea următoare trecerea în revistă a îmbunătăţirilor aduse motorului OO şi să vedeţi ce impact ar putea avea aceasta asupra aplicaţiilor dumneavoastră.

Noul model de obiecte

Întrucât această carte nu v-a prezentat programarea în PHP dintr-un punct de vedere orientat pe obiecte, această secţiune nu va intra în detalii pentru a vă prezenta exemple de modificări ale modelului de obiecte prezente în PHP 5. Pentru aceasta vă recomandăm articolul lui Harry Fuecks, *„PHP5: Coming Soon to a Webserver Near You"*, articol pe care îl puteţi citi la adresa `http://www.sitepoint.com/article/1192/`. Acest articol conţine o disecţie spectaculoasă a modificărilor în arhitectura de obiecte din PHP 5 şi modalităţile de implementare pe care le aveţi la dispoziţie.

Următoarele puncte vă oferă o trecere în revistă a modificărilor. Dacă sunteţi un dezvoltator de Java, veţi începe să observaţi similitudini puternice.

- Variabile şi metode de tip `private` şi `protected` — variabilele şi metodele de tip `private` sunt accesibile numai din cadrul clasei în care au fost declarate, în timp ce variabilele şi metodele de tip `protected` sunt disponibile şi

pentru subclasele derivate din aceasta. Această facilitate este complet nouă în PHP, aşa că nu există nici un fel de problemă de compatibilitate cu versiuni anterioare ale limbajului, alta decât că ar trebui să vă întoarceţi şi să măriţi securitatea din jurul variabilelor şi metodelor dumneavoastră.

- Introducerea metodei `_constructor()` — în capitolul 7, aţi aflat că această categorie, constructorii, poartă chiar numele clasei. În noul model este folosită metoda `_constructor()`, astfel încât dacă se întâmplă să modificaţi numele clasei să nu fiţi nevoiţi să modificaţi şi numele constructorului. Această facilitate este complet nouă în PHP, dar veţi putea în continuare să folosiţi modalitatea PHP 4 de declarare.
- Introducerea metodei `_destructor()` — această metodă asigură închiderea corespunzătoare a unui obiect şi „curăţarea" spaţiului de lucru o dată cu terminarea vieţii acestuia. Din nou, este o facilitate complet nou apărută în PHP, deşi atât în PHP 4 cât şi în PHP 5 rutinele încorporate de „garbage collection" (colectare a deşeurilor) se vor ocupa cu îndepărtarea obiectului din memorie în momentul în care acesta nu mai este activ. Metoda `_destructor()` vă permite pur şi simplu să apelaţi explicit această acţiune.
- Clase abstracte — permit unui grup de subclase să împartă o clasă părinte, fără a o utiliza direct. De exemplu, o asemenea clasă va fi folosită numai pentru a extinde alte clase şi nu va fi apelată direct. Această facilitate este introdusă pentru prima oară în PHP 5 şi, astfel, nu prezintă nici o problemă de compatibilitate cu versiuni anterioare ale limbajului, deşi veţi dori, probabil, să creaţi câteva clase abstracte acolo unde este cazul, lucru imposibil până în acest moment,
- Supraîncărcarea obiectelor (overloading) — posibilitatea de a apela metode care nu sunt declarate explicit în cadrul clasei apelate. Această facilitate este complet nouă în PHP, aşa că nu există nici un fel de problemă de compatibilitate cu versiuni anterioare ale limbajului.
- Cuvintele cheie `static`, `final` şi `constant` — folosite pentru a defini şi mai precis variabilele şi metodele din cadrul claselor dumneavoastră. Cuvântul cheie `static` permite ca variabilele şi metodele să fie accesate fără a fi necesară instanţierea clasei care le conţine. Cuvântul cheie `final` previne redefinirea de către o clasă copil a unei variabile sau a unei metode din cadrul unei clase. Cuvântul cheie `const` declară constante disponibile exclusiv în cadrul clasei respective. Aceasta este o caracteristică nouă a versiunii 5 de PHP, în ciuda faptului că, în PHP 4, existau metode ocolite de a obţine o funcţionalitate similară. Nu există probleme de compatibilitate cu versiunile anterioare, dar asta nu înseamnă că nu ar trebui să revizuiţi codul şi să introduceţi aceste cuvinte cheie acolo unde este posibil.
- Autoîncărcarea — permite să includeţi o clasă, cu ajutorul funcţiei `_autoload()`, doar atunci când respectiva clasă este instanţiată. Aceasta este o caracteristică nou introdusă în PHP 5 şi, de aceea, nu există probleme de compatibilitate cu versiunile anterioare.
- Tratarea erorilor — implementează un mecanism de tratare a erorilor similar celui prezent în limbajul Java, în timp ce tratarea erorilor în versiunea 4 de PHP depinde exclusiv de codul pe care programatorul

doreşte să-l dedice acestui proces. Tratarea erorilor se va face în PHP 5 cu ajutorul unei structuri de control de tipul `try...catch` (*încearcă... interceptează*), structură asemănătoare celei condiţionale `if...else`; scopul acesteia este de a separa datele aplicaţiei de erori. Aceasta este o caracteristică nou introdusă în PHP 5 şi, de aceea, nu există probleme de compatibilitate cu versiunile anterioare. Dacă aţi scris însă cod propriu de tratare a erorilor este o idee bună să îl convertiţi într-o structură de tipul `try...catch`.

După cum puteţi vedea, modelul de obiecte a crescut exponenţial, iar modificările sunt în mod sigur pozitive. Deşi nu este nimic în neregulă cu informaţiile fundamentale despre obiecte pe care le-aţi aflat în capitolul 7, intitulat *Lucrul cu vectorii şi obiectele*, modelul respectiv este comparabil unui automobil Dacia, în timp ce structura de obiecte implementată în PHP 5 este perfect echivalentă unui sedan de lux (fie el BMW sau Mercedes, dar, oricum, unul foarte scump).

Noi facilităţi suplimentare

Versiunea 5 de PHP va găzdui multe încă alte câteva facilităţi importante, dar, din fericire, nici o modificare nu va intra în conflict cu informaţiile pe care le-aţi acumulat pe parcursul acestei cărţi! Următoarele câteva secţiuni încearcă să vă prezinte câteva dintre aceste facilităţi noi.

SQLite

PHP 5 va conţine un motor înglobat de baze de date, pe numele său *SQLite*, ce nu va încerca să înlocuiască o bază de date robustă, fie ea MySQL, PostgreSQL sau Oracle. În schimb, această aplicaţie de baze de date simplă şi rapidă poate stoca bazele de date atât în fişiere, cât şi în memorie, fiind astfel foarte utilă în contextul unor activităţi precum stocarea datelor de sesiune şi a informaţiilor de configurare ale aplicaţiilor — în principiu, instrucţiuni simple de tipul `INSERT` şi `SELECT`. Deşi mecanismele interne ale acestui motor de baze de date sunt foarte puţin cunoscute şi discutate, bazându-ne pe datele iniţiale referitoare la viteza sa de execuţie, vă recomandăm să-l testaţi cu încredere pentru activităţi precum cele menţionate mai sus.

Implementarea XML

Versiunea 4 a limbajului PHP foloseşte mai multe librării scrise de terţe părţi pentru a controla parsarea şi interpretarea codului XML: Expat, `libxml` şi Sablotron. Expat este poate mai folosită de dezvoltatori şi este cât se poate de stabilă, dar dezvoltatorii apelează şi la celelalte două librării. Ideea este că limbajul PHP nu are, în versiunea 4, un mecanism unic, stabil şi de încredere, de parsare şi interpretare de cod XML.

O dată cu PHP 5, grupul de dezvoltatori s-a pus de acord asupra unei librării XML, librăria Gnome `libxml`, aceasta devenind fundaţia extensiilor DOM. Această librărie a fost considerată ca fiind cea mai rapidă şi mai completă şi, astfel, o fundaţie corespunzătoare pe care se poate construi funcţionalitatea XML în PHP.

Când ar trebui să trec la PHP 5?

Dacă aveţi posibilităţi de dezvoltare, puteţi să descărcaţi o versiune beta de PHP 5 şi să o instalaţi. Ca în cazul oricărei alte versiuni de tip beta, vă veţi confrunta atât cu erori, cât şi cu facilităţi încă neimplementate, ba chiar cu elemente implementate care vor dispărea însă în distribuţia finală a aplicaţiei. Dacă sunteţi intrigaţi de facilităţile enumerate în acest capitol, vă recomandăm să urmăriţi evoluţia acestei versiuni beta, instalând şi testând fiecare nouă actualizare. Veţi fi astfel complet pregătit să treceţi imediat la lcuru în PHP 5 într-un mediu de producţie în momentul în care grupul PHP consideră că a venit momentul lansării unei prime versiuni non-beta.

Compatibilitatea cu versiunile anterioare

Jurnalul de modificări şi manualul de PHP vor sublinia clar locurile în care o nouă implementare va genera probleme pentru fişierele create în versiuni anterioare, aşa că este bine să citiţi cu atenţie jurnalul de modificări, dacă nu chiar manualul, documentându-vă asupra funcţiilor dumneavoastră preferate. Filozofia din spatele PHP implică, însă, adăugarea de funcţionalitate suplimentară şi nu înlocuirea completă a unor elemente deja existente, astfel încât s-ar putea să descoperiţi că nici unul dintre scripturile dumneavoastră nu *necesită* rescrierea sa. Necesitatea rescrierii este complet diferită de rescrierea scripturilor de dragul utilizării noilor facilităţi — s-ar putea să doriţi să vă rescrieţi porţiuni de cod pentru a profita de funcţionalitatea noului model de obiecte, dar nu este obligatoriu să faceţi aceasta. În cazul în care codul dumneavoastră este de tip procedural (cum este cazul scripturilor din această carte), şansele de a nu avea nevoie de rescriere sunt mai mari de 95%.

Rezumat

Acest capitol v-a oferit o trecere sumară în revistă a funcţionalităţii nou introduse în PHP 5. V-au fost prezentate rapid adăugirile şi îmbunătăţirile aduse modelului de obiecte, precum şi alte elemente noi, cum ar fi implementarea mult îmbunătăţită a suportului de XML şi disponibilitatea unui motor integrat de baze de date, pe numele său SQLite.

Întrebări şi răspunsuri

Î. Unde aş putea citi mai multe despre PHP 5?

R. Zend Technologies, creatorul structurii Zend Engine care se află în spatele PHP, are o secţiune devotată viitorului limbajului, pe numele său *The Future of PHP* (*Viitorul limbajului PHP*). Această secţiune conţine prezentări, instrucţiuni de utilizare, articole, precum şi un forum pe care se pot schimba idei. Adresa sa Web este `http://www.zend.com/zend/future.php`. Pentru ştiri cu privire la lansările de PHP 5, urmăriţi site-ul Web PHP, la adresa `http://www.php.net/`.

CAPITOLUL 31

Facilitățile noi și compatibilitatea cu versiunile anterioare ale MySQL 4.1

În această carte v-am prezentat elementele fundamentale ale MySQL — cum să creați și să adăugați tabele, cum să adăugați înregistrări în respectivele tabele, cum să intersectați tabele și să preluați date, unele dintre funcțiile înglobate în MySQL și așa mai departe. Ați ajuns probabil și dumneavoastră la concluzia că MySQL este un sistem solid de baze de date și că execută foarte rapid majoritatea comenzilor. Aceste afirmații sunt cât se poate de adevărate. Dezvoltatorii de la MySQL AB nu s-au culcat însă pe lauri — MySQL 4.1 este următoarea versiune majoră și va include câteva îmbunătățiri care vor transforma MySQL într-un sistem și mai performant.

În acest scurt capitol, veți afla câteva informații despre facilitățile planificate spre a fi introduse în MySQL 4.1, incluzând:

- introducerea subinterogărilor și modul în care sunt acestea utilizate;
- o trecere în revistă a îmbunătățirilor din domeniul internaționalizării;
- informații despre adăugiri minore;
- informații despre următoarea versiune, MySQL 5.0.

Utilizarea subinterogărilor

Probabil că cea mai senzațională îmbunătățire adusă o dată cu versiunea 4.1 a MySQL este disponibilitatea subinterogărilor. Cu ajutorul subinterogărilor dumneavoastră aveți posibilitatea să rulați o interogare pentru a obține un set de rezultate pe care să îl folosiți ulterior drept condiție în cadrul unei alte interogări — și totul în aceeași instrucțiune. Cu alte cuvinte, aveți două instrucțiuni într-o singură interogare, a doua bazându-se pe rezultatele primei interogări.

Gândiți-vă la toate momentele din viața dumneavoastră de programator când ați ajuns în situația în care doreați să obțineți rezultate din mai multe tabele, pe care nu doreați, însă, să le intersectați, din cauza complexității lor sau, poate, pentru că nu erau în întregime relaționale. Soluția ar fi fost probabil să rulați prima interogare, după care să parcurgeți rezultatele și să rulați altă interogare.

Nu este nici o problemă cu această abordare a problemei şi, cu un cod compact, nici cantitatea de timp consumată probabil că nu ar fi o problemă. În aceeaşi situaţie însă, folosirea unei subinterogări vă va deschide uşa spre o lume nouă a eficienţei.

Exemplu de subinterogare pentru monitorizarea timpului

Să presupunem că aţi implementat un sistem care permite angajaţilor să introducă orele lucrate pentru anumite sarcini, în anumite zile, pentru o companie dată. Crearea tabelelor şi instrucţiunile de introducere de date sunt exemple rudimentare a ceea ce aţi putea întâlni într-un sistem de acest tip.

În tabela `tasks` sunt stocate numele activităţilor şi tarifele per oră corespunzătoare:

```
CREATE TABLE tasks (
    task_id tinyint PRIMARY KEY NOT NULL auto_increment,
    task_name varchar(50) NOT NULL default '',
    task_fee float(4,2) NOT NULL default '0.00'
);
INSERT INTO tasks VALUES ('1', 'Programming', '150.00');
INSERT INTO tasks VALUES ('2', 'Database Design', '200.00');
INSERT INTO tasks VALUES ('3', 'Project Management', '150.00');
INSERT INTO tasks VALUES ('4', 'Data Entry', '75.00');
INSERT INTO tasks VALUES ('5', 'Graphic Design', '150.00');
```

În tabela `employees` sunt stocate numele angajaţilor:

```
CREATE TABLE employees (
    e_id tinyint PRIMARY KEY NOT NULL auto_increment,
    e_fname varchar(50) NOT NULL default '',
    e_lname varchar(50) NOT NULL default ''
);
INSERT INTO employees VALUES ('1', 'John', 'Doe');
INSERT INTO employees VALUES ('2', 'Jane', 'Doe');
INSERT INTO employees VALUES ('3', 'Michael', 'Smith');
INSERT INTO employees VALUES ('4', 'Ralph', 'Jones');
```

În tabela `clients` sunt stocate numele clienţilor:

```
CREATE TABLE clients (
    c_id tinyint PRIMARY KEY NOT NULL auto_increment,
    c_name varchar(255) NOT NULL default ''
);
INSERT INTO clients VALUES ('1', 'ABC Company');
INSERT INTO clients VALUES ('2', 'XYZ Company');
INSERT INTO clients VALUES ('3', 'Acme Telesystems');
INSERT INTO clients VALUES ('4', 'GoNutz Inc.');
```

În cele din urmă, în tabela `billable_hours` aveţi intrările individuale. Observaţi că toate intrările din tabela `billable_hours` folosesc câmpurile ID

corespunzătoare din tabelele employee, client şi tasks pentru a crea o relaţie coerentă, individuală cu înregistrările principale ale fiecăreia.

```
CREATE TABLE billable_hours (
    bh_id int PRIMARY KEY NOT NULL auto_increment,
    work_date date,
    e_id tinyint,
    c_id tinyint,
    task_id tinyint,
    billable_time float(4,2) NOT NULL default '0.00'
);
INSERT INTO billable_hours VALUES ('', '2003-08-25', '1', '3', '2', '1.75');
INSERT INTO billable_hours VALUES ('', '2003-08-25', '1', '3', '1', '3.50');
INSERT INTO billable_hours VALUES ('', '2003-08-25', '2', '1', '3', '4.25');
INSERT INTO billable_hours VALUES ('', '2003-08-26', '2', '1', '4', '7.25');
INSERT INTO billable_hours VALUES ('', '2003-08-26', '2', '2', '1', '3.50');
INSERT INTO billable_hours VALUES ('', '2003-08-26', '3', '3', '1', '4.00');
INSERT INTO billable_hours VALUES ('', '2003-09-01', '3', '4', '2', '6.75');
INSERT INTO billable_hours VALUES ('', '2003-09-01', '3', '4', '2', '4.25');
INSERT INTO billable_hours VALUES ('', '2003-09-01', '4', '3', '3', '5.00');
INSERT INTO billable_hours VALUES ('', '2003-09-05', '4', '1', '3', '4.50');
```

Să presupunem că doriţi să afişaţi o listă de sarcini distincte efectuate de orice angajat pentru compania Acme Telesystems şi că nu folosiţi subinterogări. Pentru început, trebuie să aflaţi identificatorul companiei Acme Telesystems, după care să lansaţi o a doua interogare pentru a obţine lista de activităţi desfăşurare în cadrul acestei companii:

```
mysql> select c_id from clients where c_name = 'Acme Telesystems';
+------+
| c_id |
+------+
| 3    |
+------+
1 row in set (0.16 sec)

mysql> select distinct t.task_name as theTasks
    -> from billable_hours as bh
    -> left join tasks as t on t.task_id = bh.task_id
    -> where bh.c_id = 3;
+--------------------+
| theTasks           |
+--------------------+
| Database Design    |
| Programming        |
| Project Management |
+--------------------+
3 rows in set (0.22 sec)
```

Folosind subinterogări, aceste două instrucţiuni s-ar transforma într-una singură:

```
mysql> select distinct t.task_name as theTasks
    -> from billable_hours as bh
    -> left join tasks as t on t.task_id = bh.task_id
    -> where bh.c_id = (select c_id from clients where c_name = 'Acme
Telesystems');
```

```
+--------------------+
| theTasks           |
+--------------------+
| Database Design    |
| Programming        |
| Project Management |
+--------------------+
3 rows in set (0.22 sec)
```

În cazul de mai sus, utilizarea subinterogărilor a eliminat prima interogare, precum şi 0,16 secunde din întregul proces (plus timpul necesar tastării primei interogări!).

În general, o subinterogare este, de cele mai multe ori, o instrucţiune de tip SELECT, care nu trebuie să întoarcă mai mult de o singură coloană de rezultate. Întrucât încercaţi să faceţi corespondenţa dintre un rezultat din cea de-a doua interogare şi o condiţie din prima interogare, puteţi observa de ce o instrucţiune SELECT care ar genera mai multe coloane în a doua interogare ar duce la o eroare de corespondenţă.

Puteţi să imbricaţi oricât de multe subinterogări doriţi, atât timp cât sunt respectate regulile de bază ale interogărilor; acestea trebuie să fie valabile şi bine structurate. Uitaţi-vă la următorul exemplu, care prezintă activităţile efectuate de Michael Smith pentru GoNutz Inc.:

```
mysql> select t.task_name as theTask,
    -> bh.billable_time * t.task_fee as theTotal
    -> from billable_hours as bh
    -> left join tasks as t on t.task_id = bh.task_id
    -> where
    -> (
    -> (bh.c_id = (select c_id from clients where c_name = 'GoNutz Inc.'))
    -> and
    -> (bh.e_id = (select e_id from employees where (e_fname = 'Michael' and
e_lname = 'Smith')))
    ->);

+-----------------+---------+
| theTasks        |theTotal |
+-----------------+---------+
| Database Design |1350.00  |
| Database Design |850.00   |
+-----------------+---------+
2 rows in set (0.22 sec)
```

În următoarele secţiuni veţi afla câteva date despre unele îmbunătăţiri aduse de versiunea 4.1 a serverului de baze de date MySQL, dar probabil că nici una dintre acestea nu va avea acelaşi efect asupra stilului dumneavoastră de programare cum este posibilitatea de a folosi subinterogări în aplicaţii!

Îmbunătățiri aduse procesului de internaționalizare

O instalare standard de MySQL include deja o gamă largă de funcţii care permit folosirea de limbi străine, fie ele pe un singur octet sau multioctet, dar câteva îmbunătăţiri aduse în varianta 4.1 vă vor uşura şi mai mult implementarea.

În principiu, în MySQL 4.1 veţi putea să definiţi seturi de caractere per coloană, per tabelă şi per bază de date. Această facilitate va introduce o consecvenţă mai mare şi o flexibilitate crescută în contextul dezvoltării de aplicaţii internaţionalizate şi de site-uri Web în mai multe limbi. În plus, MySQL 4.1 implementează şi suport pentru Unicode (UTF-8).

Pentru mai multe informaţii despre îmbunătăţirile aduse lucrului cu seturile de caractere, vizitaţi manualul MySQL, care se află la adresa `http://www.mysql.com/doc/en/Charset.html`.

Îmbunătățiri suplimentare

În această secţiune veţi afla câteva date despre alte funcţii noi care vor fi implementate în versiunea 4.1 a MySQL. Aceasta nu este o listă completă a modificărilor introduse; pentru o asemenea listă, puteţi consulta jurnalul de modificări, la adresa `http://www.mysql.com/doc/en/News-4.1.x.html`.

- Creşterea vitezei. Ca şi cum MySQL nu ar fi fost deja suficient de rapid, dezvoltatorii găsesc în permanenţă căi pentru a-l face să ruleze şi mai repede.
- Crearea de tabele cu ajutorul instrucţiunii `LIKE`. Puteţi folosi o structură de genul `CREATE TABLE nume_tabela LIKE nume_alta_tabela` pentru a crea o nouă tabelă cu aceeaşi structură ca o tabelă deja existentă (dar fără date).
- Îmbunătăţiri care vizează uşurinţa în utilizare, cu referire specială la un sistem de asistenţă accesibil din linia de comandă MySQL. De exemplu, dacă tastaţi `UCASE` vă vor fi afişate instrucţiuni despre utilizarea funcţiei MySQL omonime, `UCASE()`.
- Folosirea tipurilor de date `DATE/DATETIME` se va modifica într-o oarecare măsură, dumneavoastră nemaifiindu-vă permisă introducerea datei în formatul `YYYYMMDD HHMMSS` în loc de `YYYY-MM-DD HH:MM:SS`; a devenit obligatorie folosirea tuturor separatorilor.
- `TIMESTAMP` va fi întors ca un şir în formatul `YYYY-MM-DD HH:MM:SS`, în locul valorii actuale în milisecunde.
- Vă va fi permisă specificarea unei lungimi pentru câmpurile `BLOB` şi `TEXT`, iar prin aceasta MySQL va modifica automat tipul de date în tipul corespunzător.
- Pot fi adăugate comentarii per coloană, iar în urma rulării comenzii `SHOW FULL COLUMNS FROM nume_tabela` vă vor fi afişate şi aceste comentarii.

- Vor fi adăugate extensii spaţiale, care vor crea premizele generării, stocării şi analizării unor caracteristici geografice. Puteţi citi despre aceasta la adresa `http://www.mysql.com/doc/en/Spatial_extensions_in_MySQL.html`.

Cu excepţia modificărilor legate de date calendaristice menţionate anterior, nici una dintre îmbunătăţirile enumerate aduse prin această versiune de MySQL nu o face incompatibilă cu versiunile anterioare. În jurnatul de modificări al acestor actualizări, fie ele minore sau majore, MySQL AB va publica întotdeauna date de avertizare, cu caractere aldine, indicând orice tip de incompatibilitate cauzată de vreuna dintre modificări.

Înainte, spre MySQL 5.0

Ca şi cum modificările introduse în MySQL 4.1 nu ar fi fost suficiente, dezvoltatorii lucrează din greu deja, planificând şi creând prototipuri pentru a fi incluse în MySQL 5.0. Lista facilităţilor incluse va creşte dramatic pe măsură ce dezvoltatori se vor apropia de lansarea unei versiuni alfa, dar în general vă puteţi aştepta la următoarele de la MySQL 5.0:

- Proceduri stocate, bazate pe SQL-99. Dezvoltatorii vizează de asemena posibilitatea folosirii de limbaje externe şi implementarea compatibilităţii cu limbaje precum PL/SQL şi T-SQL.
- Funcţii noi, cum ar fi suportul elementar de cursor.
- Mai mulţi paşi în sensul respectării stricte a standardelor.
- O creştere şi mai mare a vitezei de execuţie.
- Continuarea îmbunătăţirii legate de internaţionalizare şi de uşurinţa în utilizare.

Ca şi subinterogările pentru MySQL 4.1, procedurile stocate sunt Marea Realizare pe care o aşteptăm în MySQL 5.0.

Rezumat

Acest capitol v-a oferit o trecere rapidă în revistă a facilităţilor pe care le aşteptăm de la MySQL 4.1 şi chiar a ce se pare că ne va oferi MySQL 5.0, deşi MySQL 4.1 este mult mai aproape de linia orizontului. V-au fost prezentate un exemplu rudimentar de folosire a subinterogărilor şi o parcurgere a îmbunătăţirilor legate de procesul de internaţionalizare şi nu numai. Din nou, pentru a fi la curent cu evoluţia versiunii 4.1 a MySQL ar trebui să vizitaţi pagina Web `http://www.mysql.com/doc/en/News-4.1.x.html`.

Întrebări şi răspunsuri

Î. Pot să folosesc MySQL 4.1 în acest moment?

R. Deşi în momentul scrierii acestei cărţi nu a fost lansată încă o versiune alfa, MySQL 4.1 poate fi descărcat de la adresa `http://www.mysql.com/downloads/mysql-4.1.html`. Instalarea ar putea fi oarecum diferită de cea prezentată în această carte, caz în care ar trebui să citiţi instrucţiunile ataşate. De exemplu, utilizatorii de Windows nu vor avea la dispoziţie un expert pentru instalare, ci vor trebui pur şi simplu să dezarhiveze fişierul arhivă într-un director specificat în instrucţiuni.

În cazul în care descărcaţi şi instalaţi MySQL 4.1, amintiţi-vă că nu este o versiune finală — nu ar trebui niciodată să folosiţi o aplicaţie care este încă în stadiul alfa într-un mediu de producţie!

Atelier

Atelierul este conceput să anticipeze posibile întrebări, să vă ajute să recapitulaţi *cele învăţate şi să începeţi să vă puneţi cunoştinţele în practică.*

Test

1. Ce este o subinterogare?
2. Adevărat sau fals: puteţi avea un singur nivel de subinterogări.
3. Adevărat sau fals: instalarea de aplicaţii software în stadiul de dezvoltare alfa în mediul de producţie al companiei la care lucrez mă face să fiu mai interesant decât colegii mei.

Răspunsuri

1. O interogare care foloseşte setul de rezultate al unei interogări în cadrul unei instrucţiuni condiţionale aparţinând unei alte interogări.
2. Fals. Puteţi avea cât de multe niveluri doriţi, atât timp cât sunt corect scrise.
3. Fals. Vă va face mai puţin „angajat" decât pe prietenii dumneavoastră.

PARTEA a VIII-a

Anexe

Anexa A. Instalarea aplicațiilor MySQL, Apache şi PHP aflate pe CD-ROM-ul ataşat cărții 557

Anexa A
Instalarea aplicațiilor MySQL, Apache şi PHP aflate pe CD-ROM-ul ataşat cărții

Capitolele de la 1 până la 3 tratează obţinerea de pe Internet şi instalarea aplicaţiilor PHP, MySQL şi Apache, pentru a vă asigura că aveţi cea mai nouă versiune.

Dacă doriţi, însă, să porniţi cât mai repede la drum, această anexă vă va purta prin paşii instalării aplicaţiilor de pe CD-ROM-ul ataşat cărţii, atât pe un sistem Linux/Unix, cât şi pe un sistem de tip Windows.

Instalarea sub Linux/Unix

Când compilaţi aplicaţii pornind de la o arhivă sursă, procesul depinde de prezenţa pe sistemul dumneavoastră a instrumentelor de dezvoltare şi a bibliotecilor corespunzătoare.

Instrucţiunile de mai jos sunt scrise pornind de la un sistem Linux de tipul Red Hat, dar paşii de urmat sunt aceeaşi pentru a instala aplicaţiile pe orice alt sistem de Linux sau pe orice distribuţie comercială Unix.

În cazul în care, pe parcursul compilării, vă loviţi de mesaje de eroare neaşteptate, contactaţi-vă administratorul de reţea sau consultaţi documentaţia sistemului dumneavoastră de operare.

Începeţi cu drepturi de superutilizator (autentificaţi-vă ca `root` sau ca `su` şi nu ca utilizator normal de sistem) şi încărcaţi CD-ROM-ul pe sistemul dumneavoastră:

```
# mount /dev/cdrom /mnt -t iso9660
```

Instalarea serverului MySQL

Înainte de a instala serverul de baze de date MySQL trebuie să creăm pe sistemul dumneavoastră un utilizator care să ruleze serverul MySQL.

```
# /usr/sbin/groupadd mysql
# /usr/sbin/useradd -g mysql mysql
```

Vom plasa tot codul sursă care trebuie compilat în directorul `/usr/local/src`, aşa că începem prin a extrage arhiva MySQL de pe CD-ROM.

```
# cd /usr/local/src
# gunzip < /mnt/MySQL/Linux/tarballs/source/mysql-4.0.15.tar.gz | tar xf -
```

Se crează astfel un director cu numele `mysql-4.0.15` care conţine codul sursă. Treceţi în acest director şi configuraţi serverul MySQL aşa cum vi se arată în cod:

```
# cd mysql-4.0.15
# ./configure --prefix=/usr/local/mysql
```

Opţiunea de compilare `--prefix` anunţă că serverul şi clientul MySQL se vor afla în directorul `/usr/local/mysql`, director identic cu cel folosit pentru instalarea binarelor. Directorul implicit pentru instalarea pornind de la surse este de fapt `/usr/local`, aşa că ar trebui să faceţi modificarea corespunzătoare pentru a plasa MySQL în aceeaşi poziţie cu cea sugerată în capitolul 2.

O dată ce a fost rulat scriptul `configure`, putem începe compilarea lansând comanda `make`.

```
# make
```

Lansaţi comanda `make install` pentru a copia nou instalatul server MySQL, programe client şi librării în poziţia corectă pe sistemul dumneavoastră:

```
# make install
```

Următorul pas constă în crearea unui set implicit de tabele, cu ajutorul scriptului `mysql_install_db`. Aflându-vă în directorul sursă, daţi comanda:

```
# scripts/mysql_install_db
```

În continuare, trebuie să configuraţi permisiunile asupra directorului de date MySQL, pentru ca utilizatorul `mysql` pe care l-aţi creat anterior să poată rula serverul şi să poată scrie şi citi fişierele de date.

```
# chown -R mysql /usr/local/mysql/var
```

În cele din urmă, puteţi porni serverul MySQL folosind comanda `mysqld_safe` şi ar trebui să vedeţi un mesaj ca cel de mai jos.

```
# /usr/local/mysql/bin/mysqld_safe &
[1] 9678
Starting mysqld daemon with databases from /usr/local/mysql/var
```

Comanda aceasta poate fi adăugată la scriptul de pornire a sistemului dumneavoastră, asigurându-vă astfel că serverul MySQL va porni automat după restartarea calculatorului. De exemplu, pe Red Hat Linux comanda trebuie adăugată la `/etc/rc.d/rc.local`.

Dacă doriţi să vă asiguraţi că MySQL rulează, folosiţi comanda `mysqladmin status` pentru a primi un mesaj de stare scurt de la un server activ.

```
# /usr/local/mysql/bin/mysqladmin status
Uptime: 8826  Threads: 1  Questions: 10  Slow queries: 0  Opens: 8
Flush tables: 1  Open tables: 2  Queries per second avg: 0.000
```

Instalarea serverului Apache

Extrageţi codul sursă Apache de pe CD-ROM în `/usr/local/src` şi plasaţi-vă în acest director.

```
# cd /usr/local/src
# gunzip < /mnt/Apache/Linux/source/httpd-2.0.47.tar.gz | tar xf -
# cd httpd-2.0.47
```

Doriţi să instalaţi Apache în `/usr/local/apache2` şi să configuraţi serverul Web pentru a permite Dynamic Shared Objects (DSO), eliminând astfel necesitatea recompilării surselor de Apache pentru a adăuga suportul de PHP. Comanda `configure` va arăta astfel în modul următor:

```
# ./configure --prefix=/usr/local/apache2 --enable-module=so
```

În această comandă, puteţi specifica şi alte opţiuni de compilare, dar pentru scopul acestei cărţi exemplul de mai sus ar trebui să fie suficient.

Compilaţi şi instalaţi Apache, după care porniţi serverul Web cu comanda `apachectl`.

```
# make
# make install
# /usr/local/apache2/bin/apachectl start
```

Pentru a verifica dacă serverul Web rulează, deschideţi un browser Web şi introduceţi ca destinaţie numele sau adresa IP a serverului dumneavoastră. Dacă instalaţi Apache pe o staţie de lucru Linux, puteţi folosi `localhost` sau `127.0.0.1` pentru a accesa serverul. Pagina implicită a unei noi instalări (prezentată în figura A.1) vă asigură că totul funcţionează perfect.

Instalarea pachetului PHP

Extrageţi codul sursă PHP de pe CD-ROM în directorul `/usr/local/src` şi plasaţi-vă în directorul respectiv.

```
# cd /usr/local/src
# gunzip < /mnt/PHP/Linux/source/php-4.3.3.tar.gz | tar xf -
# cd php-4.3.3
```

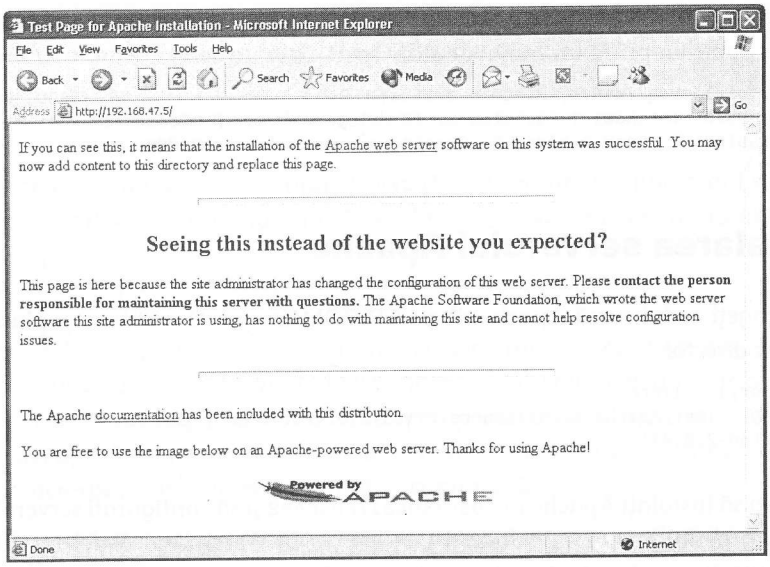


FIGURA A.1. Pagina implicită Apache.

Dorim ca limbajul PHP să poată colabora cu MySQL şi să se conecteze cu Apache prin intermediul DSO, folosind utilitarul apxs. Vom folosi, de asemenea, prefixul `/usr/local/php` pentru fişierele PHP esenţiale.

```
# ./configure  --prefix=/usr/local/php \
               --with-mysql=/usr/local/mysql \
               --with-apxs2=/usr/local/apache2/bin/apxs
```

Lansaţi apoi comenzile `make` şi `make install` pentru a compila şi a instala aplicaţia PHP.

```
# make
# make install
```

Utilitarul apxs preia majoritate sarcinilor dificile. O dată ce PHP este compilat, este copiat în poziţia corectă, iar configuraţia Apache este modificată pentru a încărca modulul la momentul pornirii.

Trebuie să mai facem încă o modificare asupra fişierului de configurare Apache pentru a instrui serverul Web să proceseze orice fişier `.php` sub forma unui script PHP. Vom activa de asemenea PHP în contextul fişierelor cu extensiile `.phtml` şi `.html`. Editaţi fişierul `/usr/local/apache2/conf/httpd.conf` şi adăugaţi-i următoarea linie:

```
AddType application/x-httpd-php .php .phtml .html
```

Reporniţi serverul Apache cu comanda `apachectl` pentru a vă asigura că se respectă modificările aduse fişierului de configurare.

```
/usr/local/apache/bin/apachectl restart
```

Pentru a verifica dacă PHP face în acest moment parte din serverul dumneavoastră Web, creaţi un script simplu cu numele `/usr/local/apache2/htdocs/index.php` care să arate în modul următor:

```
<?php
phpinfo();
?>
```

Vizitaţi în browserul dumneavoastră Web pagina `index.php` de pe noul server şi ar trebui să vedeţi o pagină care să vă ofere informaţii despre configurarea PHP, cu aspect similar celei prezentate în figura A.2.

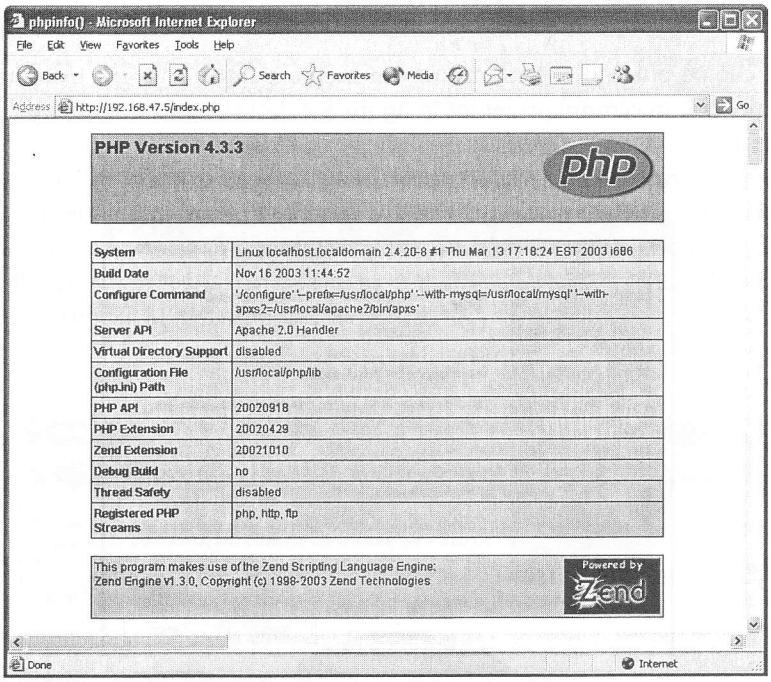


FIGURA A.2. Pagina `phpinfo()`.

Instalarea sub Windows

Fişierele incluse pe acest CD-ROM pot fi instalate pe versiunile de sistem de operare Windows 95, 98, NT, 2000, XP sau 2003. Versiunile anterioare de Windows nu permit instalarea.

Introduceţi CD-ROM-ul în calculatorul dumneavoastră şi acesta ar trebui să intre în modul autoplay. Dacă ecranul de meniu afişat în figura A.3 nu apare, daţi dublu-clic pe pictograma de CD-ROM, în fereastra My Computer.

FIGURA A.3. Meniul de instalare de pe CD-ROM.

Instalarea serverului MySQL

Dați clic pe butonul corespunzător din meniu pentru a începe instalarea de MySQL 4.0.15. În ecranul prezentat în figura A.4 dați clic pe butonul Next pentru a continua.

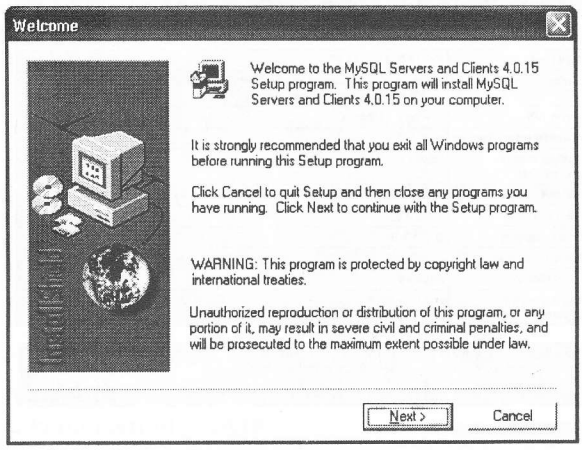


FIGURA A.4. Începutul instalării de MySQL.

Dați clic din nou pe butonul Next după ce ați citit textul de pe ecran, iar următoarea pagină vă va permite să selectați calea de instalare a fișierelor MySQL. Dacă modificați calea implicită, `C:\mysql`, trebuie să vă avertizăm că va trebui să modificați și căile de directoare din pașii următori.

Următorul pas constă în selectarea tipului de instalare — tipică, compactă, sau personalizată — așa cum vedeți în figura A.5. O instalare tipică va fi suficientă în acest context, așa că lăsați selectat elementul implicit și dați clic pe butonul Next pentru a începe procesul de instalare.

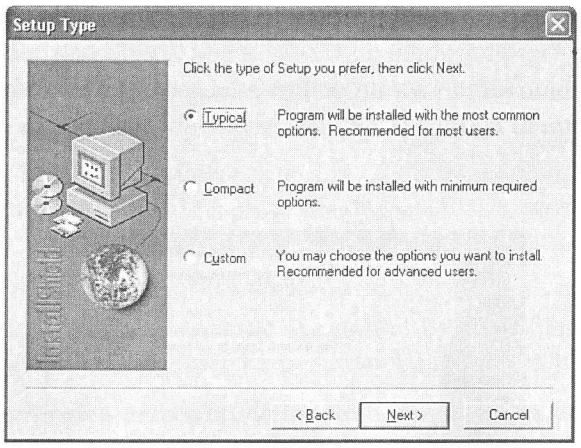


FIGURA A.5. Selectarea tipului de instalare pentru MySQL.

O dată cu finalizarea programului de instalare, mergeţi în directorul `C:\mysql\bin` (dacă aţi modificat mai sus directorul de instalare, această cale trebuie să fie relativă la directorul efectiv de instalare) şi rulaţi aplicaţia `winmysqladmin.exe`. La această primă rulare veţi fi solicitat să introduceţi un nume de utilizator şi o parolă.

După ce aţi trecut şi de asta, pictograma sub formă de semafor din bara de stare a sistemului indică starea serverului MySQL — ar trebui să fie verde. Daţi clic pe pictograma semafor şi selectaţi opţiunea Show Me (Arată-mi) pentru a lansa WinMySQLadmin, care vă va prezenta informaţiile de conectare ale serverului, cu aspect similar celor prezentate în figura A.6.

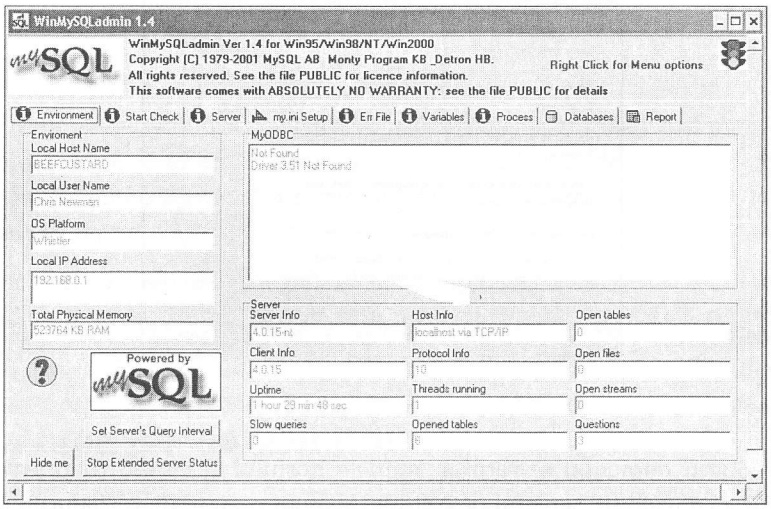


FIGURA A.6. WinMySQLadmin vă prezintă statusul serverului dumneavoastră MySQL.

Instalarea serverului Apache

Daţi clic pe butonul din meniu pentru a începe instalarea Apache 2.0.47. În ecranul prezentat în figura A.7, daţi clic pe butonul Next pentru a continua.

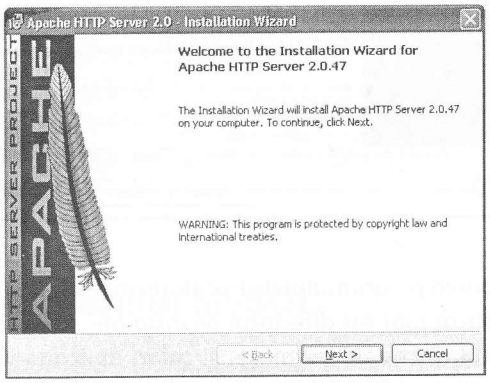


FIGURA A.7. Începerea instalării Apache.

Trebuie să fiţi de acord cu termenii licenţei, aşa cum vă sunt prezentaţi ei pe ecranul următor, pentru a putea continua instalarea, după care vă vor fi prezentate încă nişte informaţii despre versiunea pe care o instalaţi. Daţi clic pe butonul Next după ce aţi terminat de citit şi vi se va solicita introducerea informaţiilor despre server, ca în figura A.8.

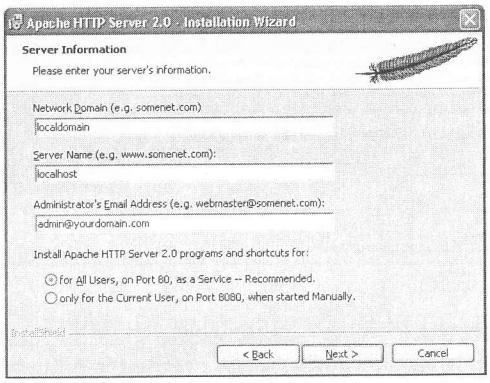


FIGURA A.8. Introduceţi detaliile serverului dumneavoastră Apache.

Introduceţi domeniul serverului, numele hostului şi adresa dumneavoastră e-mail. Dacă instalaţi serverul pe o staţie de lucru personală, introduceţi ca date ale serverului chiar `localhost` şi `localdomain`. Lăsaţi butonul radio selectat pe opţiunea implicită şi anume instalarea serverului Apache pe portul 80.

Următorul ecran vă întreabă despre tipului de instalare pe care îl doriţi, fie tipică, fie personalizată. Instalarea implicită nu are nici o problemă, aşa că nu modificaţi selectarea acestei opţiuni şi daţi clic pe butonul Next pentru a trece la pasul următor.

Veţi avea în continuare posibilitatea de a selecta fişierul destinaţie al fişierelor Apache. Calea implicită este `C:\Program Files\Apache Group`, ceea ce nu ar trebui să vă pună vreo problemă. Dacă doriţi totuşi să o modificaţi, daţi clic pe butonul Change înainde de a apăsa butonul Next.

În cele din urmă, Apache este gata să se instaleze. Un clic pe butonul Install va lansa copierea şi configurarea fişierelor pe sistemul dumneavoastră.

O dată cu terminarea procesului de instalare, serverul Apache şi programul monitor vor fi pornite, iar dumneavoastră veţi vedea o pictogramă nouă în bara de sistem. Daţi clic dublu pe această pentru a lansa aplicaţia Service Monitor, care va semăna cu imaginea prezentată în figura A.9.

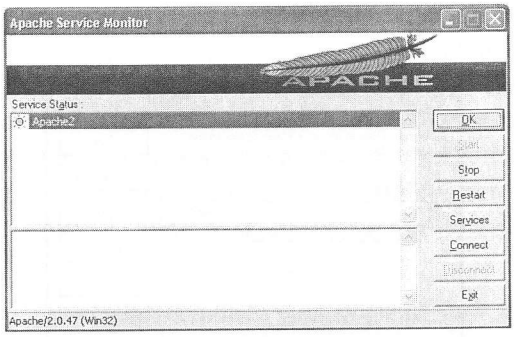


FIGURA A.9. Aplicaţia Service Monitor a serverului Web Apache.

Luminiţa verde de lângă Apache2 vă atată că serviciul rulează şi puteţi testa acest lucru deschizând un browser web şi vizitând site-ul de la adresa `http://localhost/`. Pagina implicită a serverului Apache va fi astfel afişată şi va avea aspectul din figura A.10.

Puteţi opri sau reporni serverul Web din aplicaţia Apache Service Monitor, folosind butonul din partea dreaptă. Asiguraţi-vă că serverul Apache nu rulează, înainte de a continua cu instalarea PHP.

Instalarea pachetului PHP

Daţi clic pe opţiunea corespunzătoare din meniul CD-ROM-ului pentru a instala aplicaţia PHP 4.3.3, şi se va deschide o arhivă cu autoextragere, aşa cum se vede în figura A.11.

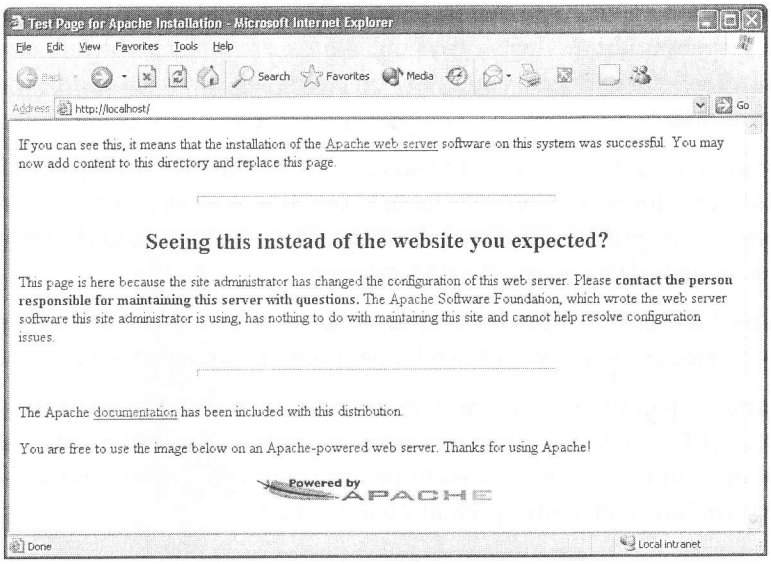


FIGURA A.10. Pagina implicită Apache pe un sistem Windows.

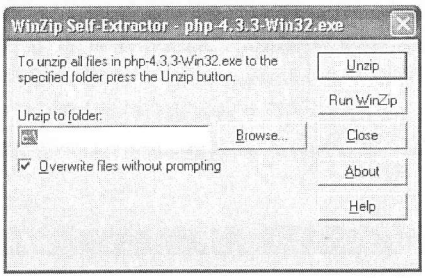


FIGURA A.11. Începerea instalării aplicației PHP.

Calea implicită de extragere este `C:\`, unde va fi creat un director nou cu numele `php-4.3.3-Win32`, aşa că ar fi preferabil să nu modificaţi această cale. Daţi clic pe butonul Extract şi fişierele vor fi plasate pe sistemul dumneavoastră. Închideţi această fereastră la terminarea procesului de extragere.

Procesul de instalare a limbajului PHP necesită în continuare intervenţia manuală pentru adăugarea modulului PHP la serverul Apache. Deschideţi directorul de configurare Apache — dacă aţi folosit calea implicită de instalare, acesta va fi `C:\Program Files\Apache Group\Apache2\conf` — şi editaţi fişierul `httpd.conf`.

Identificaţi secţiunea care conţine mai multe directive `LoadModule`, unele dintre acestea fiind precedate de prefixul `#`. Ultimele câteva linii ale secţiunii vor arăta în modul următor:

```
LoadModule userdir_module modules/mod_userdir.so
#LoadModule usertrack_module modules/mod_usertrack.so
#LoadModule vhost_alias_module modules/mod_vhost_alias.so
#LoadModule ssl_module modules/mod_ssl.so
```

Adăugaţi următoarea linie la secţiunea `LoadModule` pentru a instrui serverul Apache să încarce la pornire şi modulul PHP:

```
LoadModule php4_module c:/php-4.3.3-Win32/sapi/php4apache2.dll
```

În continuare, identificaţi secţiunea `AddType`, ale cărei directive sunt ca cele de mai jos:

```
AddType application/x-tar .tgz
AddType image/x-icon .ico
```

Adăugaţi următoarea linie pentru a instrui serverul Apache să proceseze orice fişier cu extensia `.php` ca un script PHP, extinzând directiva şi asupra fişierelor cu extensia `.phtml` şi `.html`. Puteţi adăuga şi mai multe extensii de fişiere la această linie, separându-le prin spaţii, dacă doriţi.

```
AddType application/x-httpd-php .php .phtml .html
```

Trebuie de asemenea să adăugaţi fişierul `php.ini` şi încă un fişier de tip DLL într-un director de sistem. În directorul `C:\php-4.3.3-Win32` se vor afla fişierele `php.ini-dist` şi `php4ts.dll`. Mutaţi fişierul `php.ini-dist` în directorul dumneavoastră Windows — de obiecei în `C:\WINDOWS` sau în `C:\WINNT` — redenumiţi-l sub forma `php.ini` şi mutaţi fişierul `php4ts.dll` în directorul în care sunt stocate libăriile DLL ale sistemului — de cele mai multe ori `C:\WINDOWS\SYSTEM` sau `C:\WINNT\SYSTEM32`.

Pentru a vă asigura că aceste modificări sunt implementate, reporniţi serverul Apache cu ajutorul utilitarului Apache Service Monitor. Pictograma de lângă serviciul Apache2 ar trebui să îşi modifice culoarea din roşu în verde, indicând faptul că serverul Apache a început să ruleze.

Dacă întâmpinaţi probleme la pornirea serverului Apache, verificaţi jurnalul de evenimente al sistemului din modulul Administrative Tools (instrumente de administrare) aflat în Control Panel pentru a identifica mesajul de eroare.

Sunteţi gata să verificaţi dacă modulul PHP funcţionează. În directorul `htdocs` corespunzător instalării dumneavoastră de Apache creaţi un fişier cu numele `index.php`, care să conţină următoarele linii:

```
<?php
phpinfo();
?>
```

Vizitaţi în browserul dumneavoastră Web pagina `http://localhost/index.php` de pe noul server şi ar trebui să vedeţi o pagină care să vă ofere informaţii despre configurarea PHP, cu aspect similar celei prezentate în figura A.12.

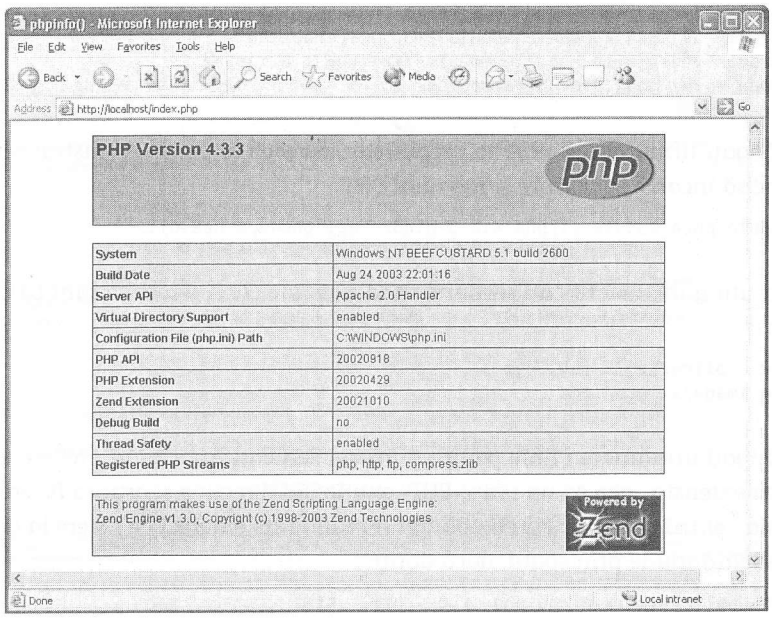


FIGURA A.12. Pagina `phpinfo()`, aşa cum apare ea pe un sistem Windows.

Rezolvarea incidentelor

Paşii descrişi în această anexă au fost testaţi cu versiunile aplicaţiilor software prezente pe CD-ROM-ul care acompaniază această carte. În cazul în care întâmpinaţi probleme la instalare, asiguraţi-vă în primul rând că aţi urmat cu exactitate paşii de mai sus. Dacă incidentele nu se rezolvă însă, următoarele site-uri Web au posibilitatea de a vă oferi asistenţă în acest sens:

`http://www.mysql.com/doc/P/r/Problems.html`

`http://httpd.apache.org/docs-2.0/faq/support.html`

`http://www.php.net/manual/en/faq.build.php`

Index

Simboluri

*/, 65
+ (adunare, operator), 80-81
& (ampersand), 85
&& (and — şi, operator logic), 85
# (diez, simbol de dispersare), 65
$ (dolar, semn), 72
" " (ghilimele duble, set de), 76
' (ghilimele simple, semn), 76
(liră sterlină, simbol), 36, 54
> (mai mare decât, operator), 84
>= (mai mare sau egal cu, operator), 84
< (mai mic decât, operator), 84
<= (mai mic sau egal cu, operator), 84
* (multiplicare, operator de), 81
! (non, operator logic), 85
!= (nonechivalenţă, operator de), 84
|| (or — sau, operator logic), 85
% (procent, semn), 154, 450
; (punct şi virgulă), 9, 44, 72, 101
- (scădere, operator), 81, 158
_ (underscore — subliniere), 72
% (wildcard — joker), 22

Ce se află pe CD-ROM

Indiferent dacă sistemul dumneavoastră de operare este Linux sau Windows, CD-ROM-ul ataşat acestei cărţi conţine un kit complet de start, care vă permite să instalaţi toate aplicaţiile software de care aveţi nevoie pentru a configura un mediu de lucru stabil, propice învăţării, testării, dar şi producerii de software:

- coduri sursă ale exemplelor prezentate în carte;
- MySQL 4.0.15;
- Apache 2.0.47;
- PHP 4.3.3.

Instrucţiuni de instalare

Conţinutul CD-ROM-ului poate fi instalat prin intermediul interfeţei sale Web, accesibilă din rădăcina CD-ROM-ului. Instrucţiunile detaliate de instalare se află în anexa A, intitulată *Instalarea aplicaţiilor MySQL, Apache şi PHP aflate pe CD-ROM-ul ataşat cărţii.*

Acord de licenţă software

O dată cu deschiderea ambalajului, vă obligaţi să acceptaţi şi să respectaţi condiţiile de mai jos.

Nu aveţi dreptul de a copia sau a redistribui în întregime conţinutul acestui CD-ROM. Copierea şi redistribuirea produselor software individuale conţinute în acest CD-ROM se supun regulilor impuse de deţinătorii drepturilor de autor ale produselor respective.

Programul de instalare de pe CD-ROM şi codul autorului (autorilor) sunt proprietatea editurii care publică acest CD-ROM şi a autorului (autorilor). Programele individuale şi celelalte elemente aflate pe CD-ROM se află sub incidenţa legilor referitoare la dreptul de autor sau sub incidenţa unui acord de licenţă de tip open-source impus de autorii acestora sau de alte entităţi deţinătoare ale dreptului de autor.

Acest pachet software este vândut sub forma prezentată, cu o garanţie strict limitată. Garanţia limitată prezentă este exclusivă şi în locul tuturor celorlalte garanţii, fie ele exprese sau implicite. Această garanţie vă oferă drepturi legale specifice. Puteţi beneficia de alte drepturi, care diferă de la ţară la ţară.

TERMENI DE UTILIZARE ŞI GARANŢIE LIMITATĂ

CITIŢI ÎNAINTE DE A DESCHIDE AMBALAJUL CD-ROM-ULUI

Prin deschiderea ambalajului sigilat acceptaţi şi vă declaraţi de acord cu termenii şi condiţiile enumerate mai jos. Dacă nu sunteţi de acord, NU DESCHIDEŢI AMBALAJUL şi prezentaţi-vă înapoi la magazin cu produsul sigilat şi cu chitanţa pentru a vă fi rambursată suma achitată pentru produs.

Acest produs software este protejat de legea proprietăţii intelectuale, iar licenţa pe care aţi achiziţionat-o vă oferă drept de folosinţă strict personală, cu excepţia cazului în care produsul v-a fost comercializat pentru a fi instalat în cadrul unei reţele. **NU** aveţi dreptul de a transfera sau a distribui acest produs software nici unei persoane prin nici un mijloc, nici măcar prin publicarea sa pe Internet. Cu excepţia unei singure copii de siguranţă, **NU** aveţi dreptul să copiaţi acest produs software sau documentaţia sa aferentă. Nu aveţi dreptul să decompilaţi, să dezasamblaţi, să modificaţi, să adaptaţi, să traduceţi sau să creaţi produse conexe acestui produs software sau documentaţiei sale. Puteţi fi tras la răspundere pentru orice copiere sau transfer care încalcă aceşti termeni de utilizare.

Acest produs software este distribuit fără garanţie, sub forma prezentă. Nici autorii, nici deţinătorii licenţei, nici Pearson Education, Inc. şi nici Grupul Editorial Corint nu fac nici o afirmaţie, fie expresă sau implicită, în privinţa acestui produs software, a calităţii, corectitudinii sau utilităţii acestuia. Nici autorii, deţinătorii licenţei, Pearson Education, Inc. şi nici Grupul Editorial Corint nu îşi asumă nici o răspundere în faţa dumneavoastră sau a oricărei alte persoane în ceea ce priveşte orice pagubă sau neplăcere cauzată sau aparent cauzată de către acest software inclusiv, dar nelimitându-se la pagubele directe, indirecte, incidentale sau pe cale de consecinţă, daunele personale, moartea din culpă, pierdere a profitului sau alte pagube din care rezultă pierdere de date, disfuncţionalităţi sau întreruperi ale fluxului de lucru. Această garanţie limitată vă oferă drepturi legale specifice. Puteţi beneficia de alte drepturi, care diferă de la ţară la ţară. Dacă produsul este defect, puteţi să îl înapoiaţi pentru a fi înlocuit.